UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

APLICACIÓN ANDROID PARA APRENDIZAJE ESPACIADO

Álvaro Ulloa Núñez
Tutor: Alejandro Sierra Urrecho

ENERO 2020

APLICACIÓN ANDROID PARA APRENDIZAJE ESPACIADO

AUTOR: Álvaro Ulloa Núñez
TUTOR: Alejandro Sierra Urrecho

Dpto. Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Enero de 2020

Resumen (castellano)

Este Trabajo Fin de Grado consiste en el desarrollo de una aplicación Android (SpacedCards) para el aprendizaje espaciado de tarjetas.

SpacedCards permite crear y editar mazos o colecciones de tarjetas de doble cara: por un lado, se escribe un objetivo de aprendizaje (una palabra en inglés, por ejemplo) y, por el otro, la referencia del objetivo (la traducción de la palabra). Muchas aplicaciones informáticas utilizan colecciones de este tipo para el aprendizaje de idiomas, por ejemplo.

SpacedCards utiliza un algoritmo de aprendizaje espaciado, SuperMemo2, para fomentar el aprendizaje. Este algoritmo determina la frecuencia con que se debe presentar una tarjeta a partir de la dificultad experimentada por el usuario con la misma. Básicamente, las tarjetas fáciles se espacian cada vez más, mientras que las difíciles se presentan con frecuencia creciente.

Las tarjetas no solo contienen texto, sino que también pueden contener imágenes, pues las imágenes son muy poderosas a la hora de evocar información y mejorar el aprendizaje.

La aplicación utiliza Firebase para el almacenamiento online de los mazos y puede ser utilizada para todo tipo de objetivos de aprendizaje: desde el aprendizaje de los nombres de un grupo de estudiantes a partir de su imagen, hasta el aprendizaje de un lenguaje de programación.

Abstract (English)

This Bachelor Thesis consists in the development of an Android application (SpacedCards) for cards spaced learning.

SpacedCards allows to create and edit decks or collections of double-sided cards: in one hand, it is written a learning objective (an English word, for example) and, on the other, the reference of the objective (the translation of the word). Many computer applications use this kind of collections for language learning, for example.

SpacedCards uses a spaced learning algorithm, SuperMemo2, to encourage learning. This algorithm determines how often a card should be shown based on the difficulty experienced by the user. Basically, easy cards are increasingly spaced, while difficult ones are presented with increasing frequency.

Cards not only contains text, but they can also contain images on them, because images are very powerful when it comes to evoking information and improving learning.

The application uses Firebase for the online storage of decks, and it can be used for every kind of learning objectives: from learning the names of a group of students, to learning a programming language.

Palabras clave (castellano)

Android, Android Studio, Kotlin, Firebase, SuperMemo2, repaso espaciado.

Keywords (English)

Android, Android Studio, Kotlin, Firebase, SuperMemo2, spaced learning.

Agradecimientos

A mi familia por todo el apoyo que me han dado siempre y en especial a lo largo de toda la carrera.

A mi tutor Alejandro, por su ayuda y por proponerme el TFG con el cuál he disfrutado mucho a lo largo del desarrollo del mismo.

Y por último a mis amigos los cuales han hecho más amena la carrera y han sido un gran apoyo durante ella.

INDICE DE CONTENIDOS

INDICE DE FIGURAS

INDICE DE CÓDIGOS

1 Introducción

En esta sección del documento se va a mostrar una introducción que nos presente un marco general del proyecto, explicando la motivación para la realización de esta aplicación, así como los objetivos que se han perseguido durante el desarrollo.

1.1 Motivación

Esta memoria de TFG tiene como propósito describir el objetivo, el desarrollo y el funcionamiento de la aplicación creada para este proyecto.

En la actualidad, es muy normal tener aplicaciones dentro de nuestros dispositivos móviles para cada uno de los ámbitos que nos podemos encontrar en nuestra vida diaria, ya sean diseñadas para aprendizaje, para entretenimiento, para gestionar nuestras finanzas, etc. Además, desde que se ha ido aumentando cada vez más el uso de internet y el almacenamiento en la nube, tenemos un mayor abanico de posibilidades con el que poder interactuar dentro de ellas.

Una de las cosas principales que buscan los usuarios al instalarse las diferentes aplicaciones es la facilidad de uso, y eso se ve muy reflejado dentro de las utilizadas para fines educativos, ya sea para centros escolares o para el aprendizaje autodidacta. Las principales aplicaciones educativas siguen este último modelo, le ofrecen al usuario una forma sencilla y adaptable al usuario para que pueda aprender sobre los temas que desee en el momento que quiera.

Actualmente tanto dentro del mercado de aplicaciones de iOS como de Android, existen una gran variedad diseñadas para el ámbito educativo como se puede ver en las siguientes gráficas recuperadas desde 42matters [1]:

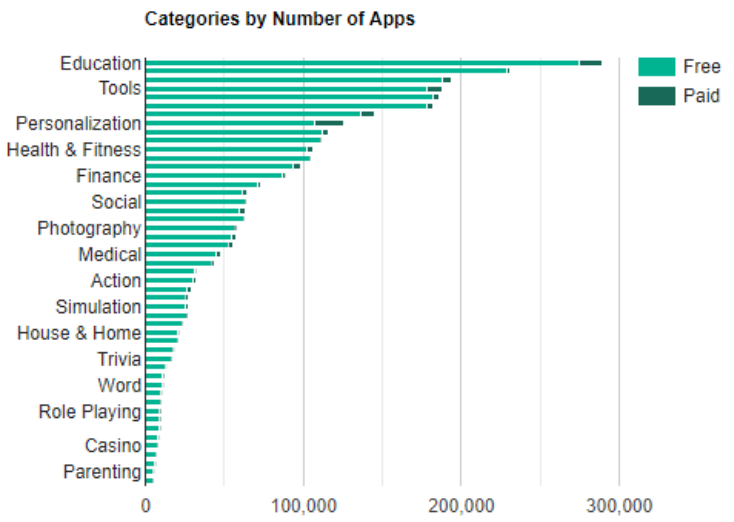

Figura 1-1: Número de aplicaciones por categoría en Google Play Store

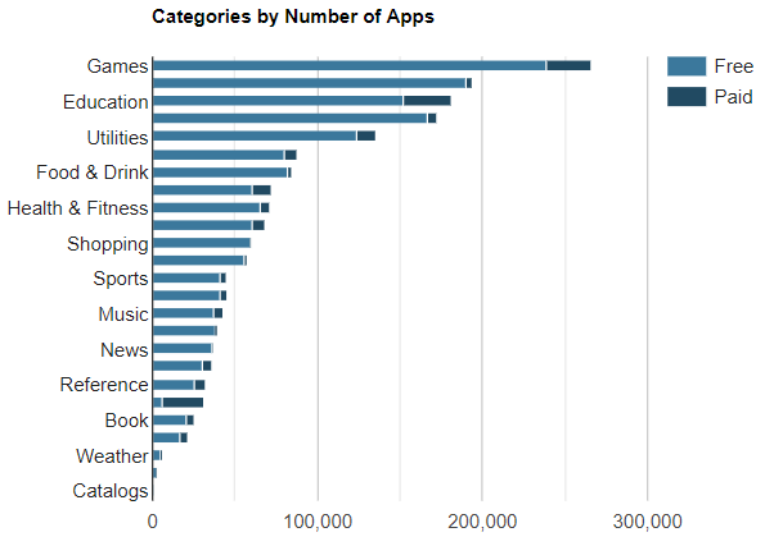

Figura 1-2: Número de aplicaciones por categoría en App Store

A pesar de ello, si comparamos con los datos obtenidos desde AppBrain [2], solo un pequeño porcentaje de ellas (6%) están superando las 50000 descargas en Google Play. Usando los datos de la página se puede dibujar la siguiente gráfica para un mejor entendimiento:

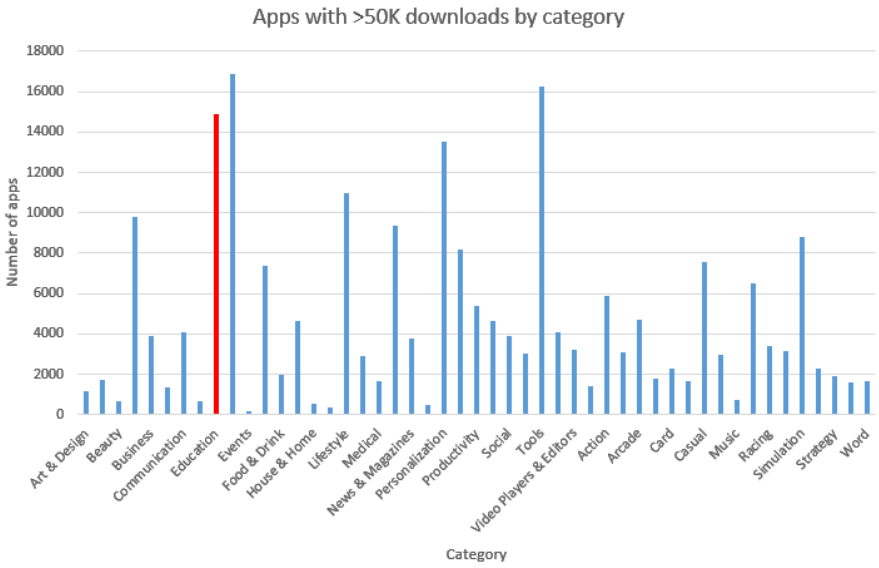

Figura 1-3: Número de aplicaciones con más de 50000 descargas por categoría en Google Play

El proyecto explicado a lo largo de esta memoria tiene su motivación inicial en la necesidad de una aplicación de uso sencillo que pudiera servir a los profesores a la hora de aprender los nombres de sus alumnos, pero sin limitarse solo a ello, sino que pudiera serle útil a más personas haciendo uso del sistema de tarjetas y mazos que la aplicación ofrece.

Existen aplicaciones que persiguen un objetivo parecido, pero o son muy complejas o todavía no tienen soporte para dispositivos móviles, que al final es lo que busca el usuario,

un pequeño software que le sirva de refuerzo, no le quite tiempo y la pueda llevar consigo a todas partes.

Como se ha comentado antes, esta aplicación no se centra en el ámbito escolar solo para encontrar a sus usuarios potenciales, cualquier persona que quiera aprender algo, por pequeño que sea, podrá hacer uso de la misma. En cuanto a los sistemas para los que estará disponible, actualmente solo está diseñada para ser ejecutada en dispositivos Android, debido a que actualmente la mayoría de usuarios tienen en su poder un producto con este sistema operativo (75,85%) y ha ido en aumento como se puede ver en la siguiente gráfica recuperada desde StatCounter [3]:

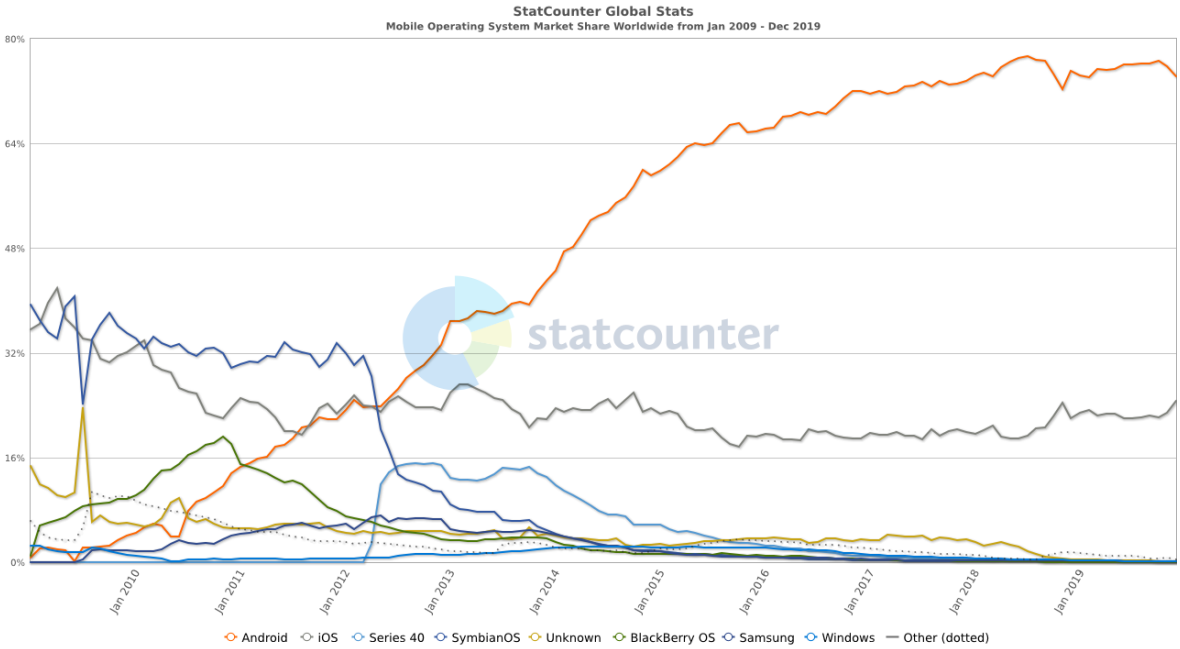

Figura 1-4: Número de dispositivos móviles operativos por sistema operativo

1.2 Objetivos

Los objetivos principales que se buscan con el proyecto son los siguientes:

O-1 – Crear una aplicación de apoyo al aprendizaje mediante un sistema de mazos y tarjetas.

O-2 – Creación y gestión de mazos y tarjetas.

O-3 – Diseño una interfaz sencilla y amigable para el usuario que le permita saber en todo momento que está haciendo y por donde moverse para realizar las tareas que desee.

O-4 – Desarrollo un sistema de refuerzo utilizando algoritmos conocidos que le sean útiles al usuario y no le quiten apenas tiempo con su uso.

O-5 – Diseño un sistema de datos y conexiones al servidor robustos que faciliten su escalabilidad y su mantenimiento.

1.3 Organización de la memoria

Este documento contiene todos los aspectos que le permitan al lector conocer las diferentes fases por las que se ha ido desarrollando el proyecto y el motivo por el que se han tomado algunas de las decisiones que han afectado en el producto final.

La memoria consta de los siguientes capítulos:

- **Estado del Arte:** Dentro de este capítulo se pondrá en contexto la aplicación frente a otros productos ya existentes explicando sus similitudes y sus diferencias con ellos.
- **Diseño:** Dentro de este capítulo se explicarán todas las funcionalidades que se diseñaron para ser cumplidas por la aplicación, el diseño de los datos para ser almacenados en la base de datos y la interfaz del usuario y el motivo de cada uno de sus elementos.
- **Desarrollo:** Dentro de este capítulo se explicará cómo funciona internamente la aplicación y se mostrará cual es el algoritmo usado y sobre que bases está diseñado.
- **Integración, pruebas y resultados:** Dentro de este capítulo se mostrará la integración con los dispositivos móviles, así como pruebas realizadas sobre la aplicación.
- **Conclusiones y trabajo futuro:** Dentro de este capítulo se darán conclusiones sobre el trabajo realizado y aspectos sobre los que poder trabajar en el futuro sobre este proyecto.

2 Estado del arte

En la actualidad existe un gran número de aplicaciones desarrolladas para su uso en dispositivos móviles, tanto móviles como tablets. Estos programas poseen una gran variedad de objetivos para el usuario, ya sea para facilitar tareas de la vida cotidiana (gestiones bancarias, correo electrónico…), para entretenimiento (plataformas de video o audio, juegos, redes sociales…), etc.

Uno de los objetivos que persiguen algunas de las aplicaciones que hay en el mercado es la formación del usuario. Existen una gran variedad de programas que a partir de diferentes métodos de aprendizaje tratan de facilitarle al usuario una forma sencilla y cómoda de estudio sobre los temas que él desee.

2.1 Repaso Espaciado

Dentro de los diferentes métodos de aprendizaje que siguen este tipo de aplicaciones, el más destacado es el método del repaso espaciado.

Siguiendo la curva del olvido de Hermann Ebbinghaus, el repaso espaciado nos aporta un método para reforzar el estudio, que consiste, a grandes rasgos, en intentar recordar lo que estamos intentando aprender justo cuando llegamos a diferentes puntos óptimos, para ralentizar el olvido del objeto en cuestión.

Uno de los primeros usos que se le dio al repaso espaciado fue la caja de Leitner, la cual consistía en una caja dividida en secciones, las cuales representaban diferentes niveles, y un conjunto de tarjetas con los contenidos que se quieren aprender. Cada día se hace repaso a diferentes niveles siguiendo un patrón y, según se haya recordado o no la tarjeta, se avanza un nivel o se devuelve al primer nivel.

Existen diferentes algoritmos que trabajan con los conceptos de la caja de Leitner, pero con algunos aspectos más sofisticados, que nos ayudan a calcular los intervalos de días, dinámicamente, en los que tenemos que volver a recordar lo que estamos intentando aprender, según las interacciones del usuario con este. Uno de esos algoritmos es SuperMemo2, el cual, a partir de cinco opciones diferentes, según la dificultad con la que el usuario ha recordado o no la tarjeta, el algoritmo va actualizando sus diferentes valores para esa tarjeta y calcula el siguiente día en el que se le tiene que volver a presentar al usuario dicha tarjeta.

2.2 Aplicaciones similares

La aplicación más conocida y usada para el aprendizaje es Duolingo. Duolingo ofrece al usuario el aprendizaje como si fuera un juego, le va dando diferentes lecciones breves sobre la materia escogida, clasificadas por niveles de dificultad, y le va recompensando con puntos y monedas cuando acierta y le resta vidas cuando falla en estas. El aprendizaje de Duolingo, además, utiliza una inteligencia artificial bastante compleja para diseñar el aprendizaje individualmente a cada usuario.

Otra aplicación bastante conocida para el aprendizaje es Anki. Anki nos ofrece el aprendizaje a partir de mazos creados por los desarrolladores o por la comunidad. Estos mazos están formados de tarjetas en las cuales por delante tienes el objetivo a aprender (ya sean palabras o imágenes) y en el reverso, el significado o traducción. El uso de estos mazos por día está limitado a un número de tarjetas fijado por el usuario por día. Además, estas tarjetas se organizan por fechas siguiendo el algoritmo SuperMemo2, mencionado anteriormente, con pequeños cambios. Este algoritmo presenta al usuario 6 botones con los que calificar la dificultad que ha tenido al recordar la tarjeta a aprender y, según la respuesta del usuario, aplica una serie de cálculos para fijar la nueva fecha sobre la que el programa te puede volver a mostrar esa tarjeta.

Una aplicación algo menos conocida es GoConqr, disponible tanto en dispositivos móviles como en versión web, aunque en un tiempo dejará de tener soporte en aplicaciones para dispositivos móviles y será solo centrado en web. GoConqr también hace uso del repaso espaciado a la hora de dividir la carga de estudios de sus fichas. Además del sistema de fichas, GoConqr cuenta con multitud de herramientas adicionales para diferentes tipos de aprendizaje, ya sean mapas mentales, apuntes, diapositivas, etc.

La aplicación sobre la que se habla en este TFG tiene similitudes con estas tres aplicaciones, ya que al igual que ellas, es una aplicación diseñada para el aprendizaje y se basa en el uso de tarjetas para ello. Al igual que con Anki, y a diferencia de Duolingo, esta aplicación hace uso del algoritmo SuperMemo2 para dirigir el aprendizaje del usuario sobre los diferentes mazos, pero presenta diferencias al hacer uso de una versión simplificada, reduciendo el número de opciones con las que el usuario puede reaccionar a la dificultad de la tarjeta dejándolo en 3 botones. Además, Anki pone un límite en el número de tarjetas a mostrar por día, mientras que en esta aplicación no hay límite, el usuario podrá repasar todas las tarjetas que tenga disponibles en esa fecha y podrá cortar el intento en el momento que desee. Y con GoConqr, la mayor diferencia es el objetivo de la aplicación, ya que en GoConqr las tarjetas son solo una de una gran cantidad de herramientas que nos ofrece la aplicación, mientras que en este TFG se ha desarrollado toda la aplicación alrededor del repaso espaciado mediante el uso de tarjetas.

3 Diseño

Dentro de este capítulo de la memoria se va a explicar todo el proceso de diseño de la aplicación, tanto a la hora de especificar los requisitos que esta debe de cumplir, como a tratar los aspectos más visuales de la misma.

Como se ha mencionado previamente, SpacedCards es una aplicación diseñada para el uso de repaso espaciado a través del uso de mazos de tarjetas creadas y personalizadas por el propio usuario.

Estas tarjetas son de doble cara, lo que quiere decir que el usuario podrá introducir información en ambas caras de cada una de las tarjetas, donde la tarjeta frontal contendrá la información que el usuario desea aprender y la cara trasera contendrá la referencia relacionada con la parte frontal. Veamos un ejemplo:

El usuario desea aprender un nuevo idioma y empieza a crear su propio mazo de tarjetas dentro de la aplicación. Al crear cada una de las tarjetas que contendrá el mazo, en la parte frontal introducirá la palabra o expresión en el idioma que desea aprender y en la parte trasera, la palabra o expresión en su lengua materna o que le sea más cómoda para usar de referencia.

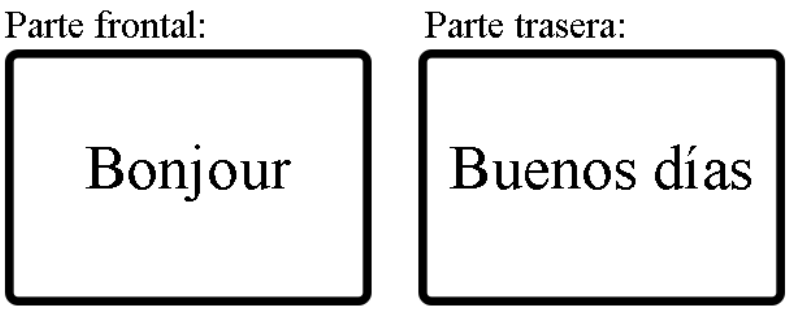

Figura 3-1: Ejemplo de tarjeta con solo texto

Este sistema es muy fácil de entender y de llevar a la práctica por parte de cualquier usuario. Como solo con texto hay algunos temas que podría resultar difícil de expresar la información a introducir en las tarjetas, la aplicación también permite que el usuario pueda escoger imágenes de su galería para colocar en la parte frontal de la tarjeta en lugar del texto, dando así la posibilidad de generar una mayor variedad de ellas con esta nueva funcionalidad.

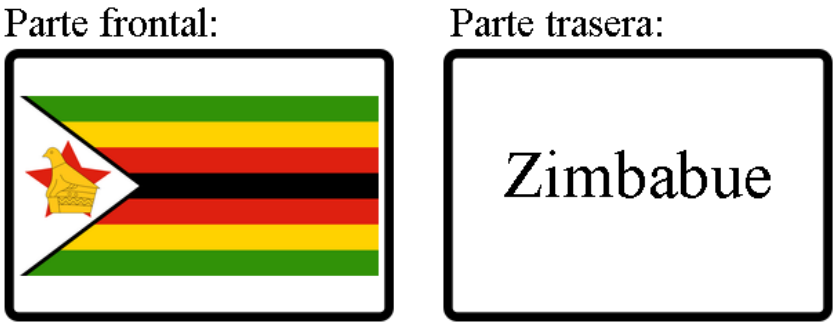

Figura 3-2: Ejemplo de tarjeta con imagen y texto

3.1 Subsistemas

En primer lugar, se van a mostrar y explicar los diferentes subsistemas que componen la aplicación. Cada uno de estos subsistemas están definidos para desempeñar funciones concretas y así separar mejor los requisitos del proyecto para su mejor comprensión. Cada sistema a su vez puede estar compuesto de otros subsistemas que especifiquen mejor las acciones que deben realizar.

Los diferentes sistemas que componen la aplicación son los siguientes:

3.1.1 Sistema de usuarios

El sistema de usuarios engloba toda la funcionalidad referente al manejo y creación de los datos de los diferentes usuarios dentro de la aplicación. Es el sistema principal de la aplicación, ya que es el que le permite al usuario acceder a la ella y está íntimamente relacionado con la parte de los mazos.

3.1.1.1 Subsistema de registro

El subsistema de registro es el que contiene toda la funcionalidad de la aplicación referente a la creación de los usuarios y de su entrada dentro de la base de datos de autenticación de Firebase. El registro se realizará mediante un correo electrónico válido y una contraseña con un número mínimo de caracteres.

3.1.1.2 Subsistema de inicio de sesión

El subsistema de inicio de sesión contiene la funcionalidad que le permite al usuario iniciar sesión dentro de la aplicación y poder recuperar sus datos para ser mostrados dentro de su página personal.

3.1.1.3 Subsistema de cambio de contraseña

El subsistema de cambio de contraseña contiene la funcionalidad necesaria para permitirle al usuario solicitar un correo electrónico a la dirección que indique, para poder cambiar la contraseña de su cuenta en caso de que se le olvide o desee modificarla.

3.1.2 Sistema de mazos

El sistema de mazos es otro de los sistemas principales de la aplicación ya que es el que engloba la funcionalidad que le va a permitir al usuario crear mazos y gestionarlos para su uso en los repasos que se hagan para su aprendizaje.

3.1.2.1 Subsistema de creación de mazos

El subsistema de creación de mazos contiene la funcionalidad necesaria para que el usuario, a partir de una cadena de texto que contenga el nombre del mazo, pueda crear un mazo en la aplicación y actualice la base de datos para añadirlo en su entrada correspondiente dentro de la base de datos de Firebase.

3.1.2.2 Subsistema de gestión de mazos

El subsistema de gestión de mazos contiene la funcionalidad para mostrarle al usuario el contenido de los diferentes mazos que haya creado y poder eliminarlos como acceder a sus tarjetas para realizar las diferentes gestiones que desee con ellas.

3.1.2.3 Subsistema de repaso de mazos

El subsistema de repaso de mazos contiene la funcionalidad que le permite al usuario realizar el repaso de las tarjetas de sus diferentes mazos. Además de mostrarle al usuario diferente información previa al comienzo de su repaso, este sistema actualiza la información de los elementos dentro de la base de datos tras la finalización de los diferentes repasos que haga el usuario sobre sus mazos.

3.1.3 Sistema de tarjetas

El sistema de tarjetas es el último de los sistemas principales que componen la aplicación, es el sistema que engloba toda la funcionalidad necesaria para permitirle al usuario crear y gestionar las tarjetas de los diferentes mazos que haya ido creando a lo largo del uso de la aplicación.

3.1.3.1 Subsistema de creación de tarjetas

El subsistema de creación de tarjetas contiene la funcionalidad para permitirle al usuario crear las diferentes tarjetas que usará dentro de los mazos para aprender sobre las materias o temas que desee. Se le permitirá crear tanto tarjetas de texto como con imágenes.

3.1.3.2 Subsistema de gestión de tarjetas

El subsistema de gestión de tarjetas contiene la funcionalidad para que el usuario pueda eliminar y editar las diferentes tarjetas que haya creado. Dentro de la edición se le permitirá cambiar el tipo de tarjeta entre texto e imagen.

3.1.3.1 Subsistema de subida de imágenes

El subsistema de subida de imágenes contiene la funcionalidad necesaria para la subida de imágenes desde la galería del teléfono a la aplicación para su uso dentro de algunas de las tarjetas que componen los mazos del usuario. Esta funcionalidad se encuentra tanto dentro de la creación de las tarjetas como en su edición, por eso está separada en un subsistema distinto.

3.2 Requisitos funcionales

En esta sección se van a definir los diferentes requisitos funcionales que debe cumplir la aplicación. Todos los requisitos van a ser agrupados siguiendo los diferentes sistemas y subsistemas que se han definido a lo largo del punto 3.1.

3.2.1 Sistema de usuarios

3.2.1.1 Subsistema de registro

RF-1 – Registro: El sistema debe permitir a cualquier usuario que use la aplicación la creación de un usuario dentro del sistema tras rellenar el formulario correspondiente.

RF-2 – Correo electrónico: El sistema debe hacer la comprobación de que el usuario ha introducido un correo electrónico válido para el registro.

RF-3 – Contraseña: El sistema debe hacer la comprobación de que el usuario ha introducido una contraseña con una longitud superior a la mínima especificada.

RF-4 – Cuenta única: El sistema debe hacer la comprobación con la base de datos de que el correo electrónico ya esté asociado o no a una cuenta para evitar que existan varias con el mismo correo.

3.2.1.2 Subsistema de inicio de sesión

RF-5 – Inicio de sesión: El sistema debe permitir a cualquier usuario que posea una cuenta dentro del sistema a iniciar sesión con su correo electrónico y contraseña tras comprobar que sean válidas.

RF-6 – Correo electrónico: El sistema debe hacer la comprobación de que el usuario ha introducido un correo electrónico válido para el inicio de sesión.

RF-7 – Contraseña: El sistema debe hacer la comprobación de que el usuario ha introducido una contraseña con una longitud superior a la mínima especificada.

3.2.1.3 Subsistema de cambio de contraseña

RF-8 – Solicitud correo: El sistema debe permitir al usuario solicitar un correo de cambio de contraseña el cual solo se enviará en caso de que exista una cuenta con el correo electrónico especificado.

RF-9 – Correo electrónico: El sistema debe hacer la comprobación de que el usuario ha introducido un correo electrónico válido para la solicitud del correo de cambio de contraseña.

3.2.2 Sistema de mazos

3.2.2.1 Subsistema de creación de mazos

RF-10 – Creación de mazo: El sistema debe permitir al usuario crear un nuevo mazo vacío a partir del nombre que indique.

RF-11 – Nombre único: El sistema debe hacer la comprobación de que el usuario no tenga ya un mazo con el nombre indicado para evitar conflictos.

RF-12 – Comprobación texto creación: El sistema debe comprobar que el texto introducido por el usuario para el nuevo mazo no esté vacío, en cuyo caso mostrará un mensaje de error.

3.2.2.2 Subsistema de gestión de mazos

RF-13 – Edición de mazo: El sistema debe permitir al usuario editar cualquiera de los mazos que haya creado previamente.

RF-13.1 – Edición del nombre del mazo: El sistema debe permitir al usuario editar el nombre del mazo desde la lista principal del usuario.

RF-13.2 – Edición de tarjetas del mazo: El sistema debe mostrar al usuario una lista de las tarjetas creadas dentro del mazo para poder gestionarlas

RF-14 – Comprobación texto edición: El sistema debe comprobar que el texto introducido por el usuario para mazo no esté vacío, en cuyo caso mostrará un mensaje de error.

3.2.2.3 Subsistema de repaso de mazos

RF-15 – Repaso del mazo: El sistema debe permitir al usuario realizar un repaso de las cartas que tenga disponibles para ese día de cada mazo que tenga creado.

RF-16 – Fecha de último repaso del mazo: El sistema debe mostrar al usuario la fecha en la que realizó el último repaso del mazo antes de iniciar uno nuevo.

RF-17 – Tarjetas disponibles: El sistema debe mostrar al usuario el número de tarjetas que tiene disponibles para el día actual.

RF-18 – Botón de comienzo del repaso: El sistema debe mostrar u ocultar el botón de comienzo del repaso tras comprobar si el usuario tiene o no cartas disponibles para repasar en el día actual.

RF-19 – Terminar repaso: El sistema debe permitir al usuario terminar su repaso actual en cualquier momento guardando así su progreso actual.

3.2.3 Sistema de tarjetas

3.2.3.1 Subsistema de creación de tarjetas

RF-20 – Creación de tarjetas de solo texto: El sistema debe permitir al usuario crear una nueva tarjeta dentro de un mazo que contenga texto tanto en su parte frontal como en su reverso. Ambos textos vendrán introducidos por el usuario.

RF-21 – Creación de tarjetas con imagen: El sistema debe permitir al usuario crear una nueva tarjeta dentro de un mazo que contenga una imagen en su parte frontal y texto en el reverso.

RF-22 – Comprobación textos creación: El sistema debe comprobar que ninguno de los textos introducidos para la creación de la tarjeta esté vacío, en cuyo caso se mostrará un mensaje de error.

3.2.3.2 Subsistema de gestión de tarjetas

RF-23 – Edición de tarjetas: El sistema debe permitir al usuario poder editar cada una de las tarjetas que haya creado. Cada tarjeta se podrá editar de modo que se le permita cambiar el tipo de carta en cualquier momento.

RF-23.1 – Edición de tarjetas de solo texto: El sistema solicitará al usuario el texto frontal y el reverso y tras guardar, la carta se actualizará con la nueva información.

RF-23.2 – Edición de tarjetas de solo texto: El sistema solicitará al usuario una imagen de la galería y el texto del reverso y tras guardar, la carta se actualizará con la nueva información.

RF-24 – Comprobación textos edición: El sistema debe comprobar que ninguno de los textos introducidos para la creación de la tarjeta esté vacío, en cuyo caso se mostrará un mensaje de error.

3.2.3.3 Subsistema de subida de imágenes

RF-25 – Subida imagen desde galería: El sistema debe permitir al usuario escoger una imagen de su galería e importarla a la aplicación para ser usada en las tarjetas.

RF-26 – Solicitud de permisos de acceso: El sistema deberá pedir al usuario los permisos correspondientes para acceder a su galería para poder subir imágenes. En caso de que se rechace, el mensaje seguirá saliendo hasta que se acepte y no permitirá al usuario subir imágenes.

RF-27 – Tratamiento de imágenes: El sistema debe someter las imágenes subidas a un tratamiento en el cual se redimensionen para que ocupen lo mínimo posible sin afectar a la visualización y reconocimiento de estas.

3.3 Requisitos no funcionales

Aparte de los requisitos funcionales que tiene que cumplir nuestra aplicación, también se han definido una serie de requisitos no funcionales que ha de cumplir, que son los siguientes:

RNF-1 – Interfaz sencilla: El sistema debe de tener una interfaz de cara al usuario sencilla, de modo que en todo momento el usuario se encuentre cómodo usando la aplicación.

RNF-2 – Interfaz adaptable: El sistema debe proporcionar soporte para diferentes tamaños de pantalla, de modo que sus vistas se adapten correctamente a cada dispositivo móvil.

RNF-3 – Tiempo de respuesta: El sistema debe proporcionar un tiempo de respuesta rápido a las transiciones entre vistas y la recogida de datos del servidor.

RNF-4 – Consumo de datos: Dado que la aplicación realiza una gran cantidad de descargas y subidas de las tarjetas y mazos al servidor, el sistema debe reducir el coste de estas acciones para reducir el consumo de datos del usuario en el mayor modo posible.

RNF-5 – Seguridad y privacidad: El sistema debe proporcionar seguridad y privacidad a los datos de cada usuario. Tanto con sistemas de autenticación como diseñando los datos de modo que se eviten ciberataques comunes que puedan comprometer estos datos.

3.4 Base de Datos

Para almacenar toda la información de los usuarios, tanto sus datos de inicio de sesión como sus mazos y tarjetas, se ha hecho uso de la plataforma de desarrollo Firebase proporcionada por Google.

Entre la gran variedad de servicios que nos proporciona Firebase, se ha hecho uso principalmente de Realtime Database (para la gestión de los datos) y Authentication (para gestión de los usuarios).

3.4.1 Realtime Database

Realtime Database nos permite mantener y gestionar una base de datos en la nube dentro del servidor que Firebase nos proporciona.

Esta base de datos no es relacional, es una base de datos NoSQL, dado que toda la información que se introduce en la base de datos va dentro de un gran fichero JSON y por lo tanto tiene diferentes optimizaciones y funcionalidades.

Los ficheros JSON son muy usados a la hora de transmitir información, tanto dentro de aplicaciones como entre aplicaciones y servidores. Poseen una sintaxis muy simple y clara que permite entender fácilmente la información que contienen, ya que todos sus datos vienen dados por un par clave-valor. Por este mismo motivo podemos encontrar muchas veces un fichero JSON dentro de otro JSON.

En el caso de esta aplicación, tanto para poder trabajar entre las diferentes actividades del programa como para enviar datos con el servidor y descargarlos, se ha utilizado también el formato JSON en todo momento para poder transmitir la información de una manera cómoda.

En el caso de las tarjetas la estructura que se ha seguido es la siguiente:

Card {
    textFace: String
    textReverse: String
    image: Boolean
    imageB64: String
    easiness: Double
    interval: Integer
    quality: Integer
    repetitions: Integer
    nextPracticeDay: Integer
    nextPracticeMonth: Integer
    nextPracticeYear: Integer
    onDate: Boolean
}

- **textFace:** Texto frontal con la información que va a tener la tarjeta.
- **textReverse:** Texto trasero con la referencia que va a tener la tarjeta.
- **image:** Flag que indica si en la parte frontal de la tarjeta se utiliza un texto o una imagen. Está puesto para poder indicar mejor que se va a pintar en cada pantalla de la aplicación.
- **imageB64:** cadena que contiene la imagen codificada en Base 64.
- **easiness:** Valor calculado por el algoritmo SuperMemo2 que indica la dificultad de recordar esa tarjeta por el usuario.

- **interval:** Número de días añadidos a la fecha del último recuerdo de la tarjeta para determinar el siguiente día a mostrar esa tarjeta. Calculado por el algoritmo SuperMemo2.
- **quality:** Valor seleccionado por el usuario al recordar la tarjeta. Indica el grado de dificultad al recordar la tarjeta y será usado por el algoritmo SuperMemo2.
- **repetitions:** Número de veces seguidas en las que el usuario no ha podido recordad fácilmente la tarjeta.
- **nextPracticeDay:** Día del mes en el que se volverá a mostrar la tarjeta al usuario.
- **nextPracticeMonth:** Mes en el que se volverá a mostrar la tarjeta al usuario.
- **nextPracticeYear:** Año en el que se volverá a mostrar la tarjeta al usuario.
- **onDate:** Flag que indica si la tarjeta se encuentra en una fecha disponible para ser mostrada.

Los campos de la tarjeta especificados para las fechas están declarados por separado como diferentes Integers, en vez de como algún tipo ya definido, como puede ser el caso de Calendar ofrecido por la librería de java.util, ya que, al trabajar con ellos, existía un problema de compatibilidad con los ficheros JSON que se enviaban al servidor de Firebase.

Por el mismo motivo que con las fechas, las imágenes deben tener un tratamiento especial para poder trabajar con ellas en los ficheros JSON que hace uso la aplicación. Para ello es necesario codificarla en Base 64 para poder trabajar con ellas como un String.

En el caso de los mazos la estructura es la siguiente:

Deck {
- **deckName**: String
- **lastPracticeDay**: Integer
- **lastPracticeMonth**: Integer
- **lastPracticeYear**: Integer
- **cards**: ArrayList<Card>

}

- **deckName:** Nombre del mazo.
- **lastPracticeDay:** Día del mes en el que fue el último repaso del mazo.
- **lastPracticeMonth:** Mes en el que fue el último repaso del mazo.
- **lastPracticeYear:** Año en el que fue el último repaso del mazo.
- **cards:** Conjunto de tarjetas que contiene el mazo.

Al igual que pasa con las tarjetas, por motivos de compatibilidad al usar los ficheros JSON, el formato de las fechas está definido como tres variables Integer.

Como los mazos son personales para cada usuario, existe una estructura por encima de los mazos que son los usuarios y cuya estructura es la siguiente:

UID {
- **decks:** ArrayList<Deck>
- **name:** String

}

- **UID:** Dentro del JSON, cada entrada de un usuario usa como clave su UID proporcionado por la herramienta de autenticación de Firebase.
- **decks:** Conjunto de mazos que posee el usuario.
- **name:** UID del usuario, usado internamente en la aplicación para poder acceder a su entrada del JSON de la base de datos.

Y, por último, la primera entrada del JSON que engloba todo lo mencionado anteriormente dentro de la base de datos es la estructura general de usuarios:

Usuario {
- **users**: ArrayList<User>

}
- **users:** Conjunto de usuarios de la aplicación

3.5 Pantallas

A continuación, se van a mostrar y explicar las diferentes pantallas de la aplicación, así como los diferentes resultados al interactuar con los diferentes elementos que las componen. Además, se comentará un poco el ciclo de navegación entre las diferentes pantallas para entender la estructura de la aplicación.

3.5.1 Pantalla de login

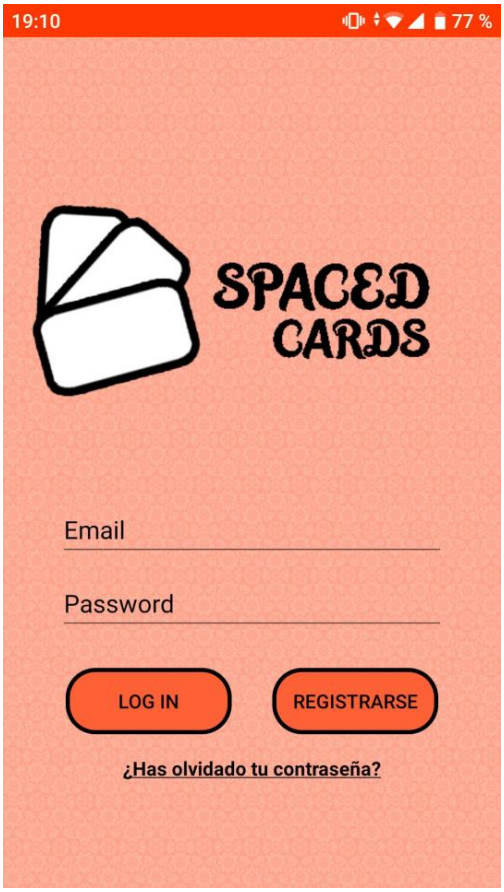

Figura 3-3: Pantalla de login

La primera pantalla que se muestra nada más abrir la aplicación es la pantalla de login. Desde esta pantalla el usuario tiene acceso a todas las funcionalidades implementadas para el servicio de autenticación proporcionado por Firebase.

La pantalla cuenta con un par de campos de texto donde el usuario podrá introducir sus datos, tanto para inicio de sesión como para registrarse en la aplicación. Estos campos de texto mostrarán mensajes al usuario en caso de que alguno de los campos no cumpla los requisitos establecidos para los datos del usuario.

En el caso de que decida registrarse, al introducir su correo y la contraseña, tras darle al botón de registrarse, el sistema creará su entrada en la base de datos y le redirigirá a su pantalla personal donde se le mostrarán su lista de mazos.

En el caso de que decida iniciar sesión, el usuario deberá introducir correctamente su correo y su contraseña, y deberá darle al botón de Log In. Si los datos introducidos son correctos, la aplicación le redirigirá a su pantalla personal con su lista de mazos.

En el caso en el que el usuario haya olvidado la contraseña asociada a su correo, existe la opción de darle al texto debajo de los dos botones principales, el cual le mostrará el siguiente diálogo:

Figura 3-4: Diálogo para cambio de contraseña

Al introducir el usuario su correo electrónico y darle a enviar, el sistema le enviará un correo donde vendrá incluido la dirección a la que debe acceder para cambiar la contraseña:

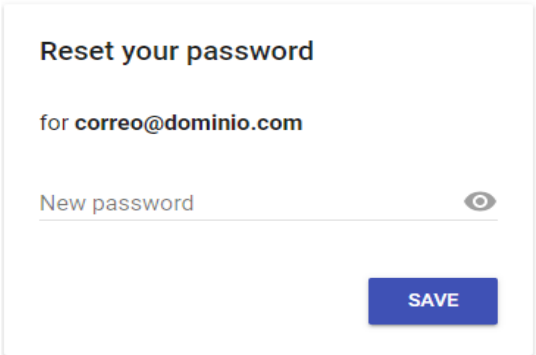

Figura 3-5: Formulario cambio de contraseña

Tras introducir la nueva contraseña, al darle al botón de guardar, el usuario ya podrá iniciar sesión con la contraseña actualizada.

3.5.2 Pantalla de listado de mazos

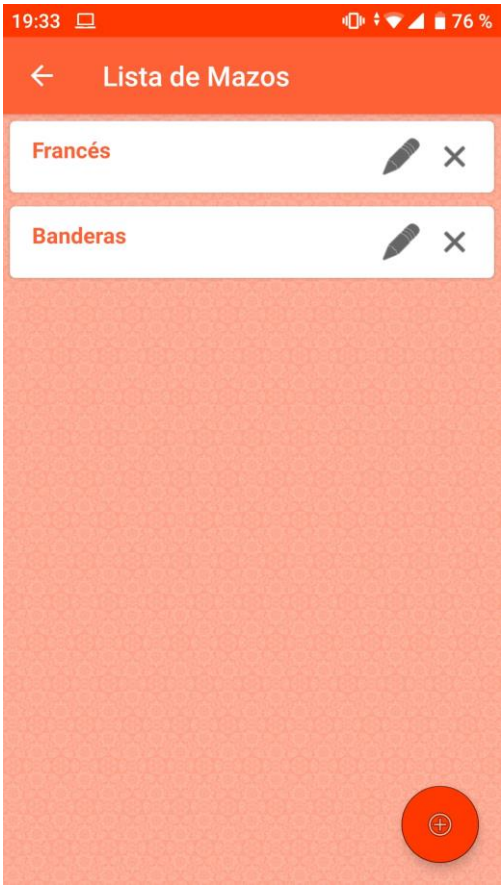

Figura 3-6: Pantalla de listado de mazos del usuario

La pantalla de listado de mazos va a ser la pantalla principal para el usuario dentro de la aplicación. Tras haber iniciado sesión, será redirigido a esta pantalla donde tendrá acceso a su listado de mazos, tendrá opción de crear nuevos y podrá cerrar sesión si desea iniciar con otra cuenta nueva.

El listado de mazos está diseñado con una vista recicladora, de modo que en el momento en el que se actualice la lista añadiendo, borrando o editando mazos, la propia vista hará su actualización dejando reflejados los cambios en la vista instantáneamente. Además, para casos en los que la lista de mazos sea extensa, la propia vista cuenta con scroll vertical para poder ver la lista completa.

Cada elemento de la vista recicladora es propio para cada mazo del usuario y el usuario puede realizar tres acciones con él:

- Si el usuario pulsa el nombre del mazo será redirigido a una pantalla resumen del mazo.
- En caso de que pulse el símbolo del lápiz, el usuario será llevado a la pantalla de edición del mazo.
- Pulsando la “X” al usuario le aparecerá el siguiente diálogo para confirmar la eliminación del mazo:

Figura 3-7: Diálogo para confirmación de eliminación del mazo

Otra de las opciones que hay dentro de esta pantalla es la de crear un nuevo mazo. Para ello, el usuario deberá de pulsar en el botón flotante de la parte inferior derecha de la pantalla y le aparecerá el siguiente diálogo de creación:

Figura 3-8: Diálogo para creación de un mazo

En caso de que el usuario no introduzca ningún nombre, o este ya haya sido utilizado por otros de sus mazos, la aplicación mostrará unos mensajes de error en la parte inferior de la pantalla. En caso de que el nombre sea válido, la lista se actualizará añadiendo el nuevo mazo y, por lo tanto, la vista recicladora también.

Por último, si el usuario pulsa la flecha en la parte superior izquierda, será redirigido a la pantalla de login, habiendo cerrado su sesión previamente.

3.5.3 Pantalla de edición de mazos

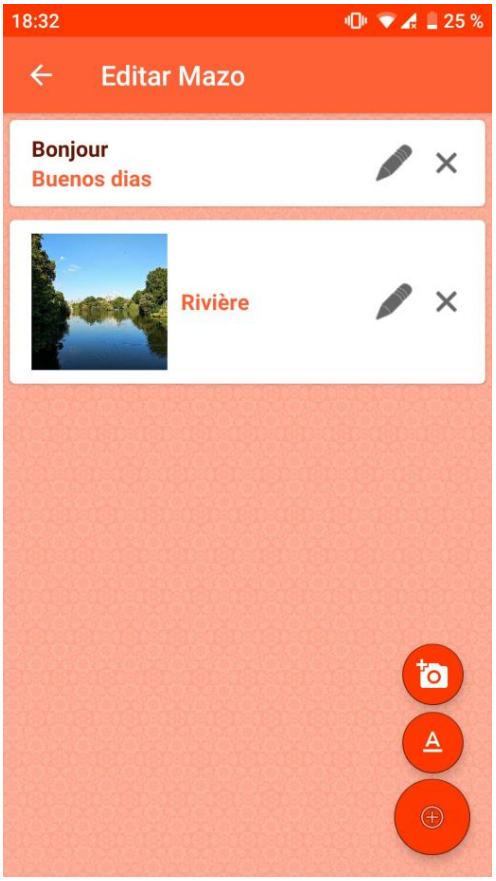

Figura 3-9: Pantalla de listado de tarjetas del mazo

Esta pantalla es muy similar en aspecto a la pantalla principal del listado de mazos del usuario. Esto es debido a que ambas comparten una vista recicladora como elemento principal de la vista para contener la información.

En este caso dentro de la vista podemos encontrar dos tipos de elementos similares, pero que difieren en la forma de presentar la información de la tarjeta, dependiendo de si es una tarjeta de solo texto o si contiene imágenes:

- En el caso de que la tarjeta solo contenga texto, se mostrará el texto frontal primero y el trasero debajo, con una diferencia de color entre ambos para poder diferenciarlos más fácilmente.
- En el caso de que contenga una imagen en la parte frontal, se mostrará la imagen y a su derecha el texto trasero. Están puestos en esa disposición para intentar ahorrar espacio dentro de la vista recicladora y que no queden muy grandes en comparación con los que son solo texto. Además, la imagen está puesta en un tamaño pequeño pero que se pueda llegar a ver lo que representa.

Con estos elementos, el usuario puede interactuar solo con los dos botones de la derecha, tanto el del lápiz para ser redirigido a la pantalla de edición de tarjeta, como con la “X” para que aparezca el diálogo de borrado de la tarjeta.

Figura 3-10: Diálogo para confirmación de eliminación de la carta

En la parte inferior derecha ahora el usuario cuenta con 3 botones distintos:

- El más grande, con el símbolo del "+". Cada vez que sea pulsado iniciará una animación desplegando o replegando los otros dos botones fuera o dentro de él.
- El botón con la cámara mostrará al usuario un diálogo desde el cual le permitirá añadir una tarjeta que contenga una imagen en la parte frontal.
- El botón con la "A" subrayada le permitirá al usuario crear una tarjeta que solo contenga texto en ambas caras.

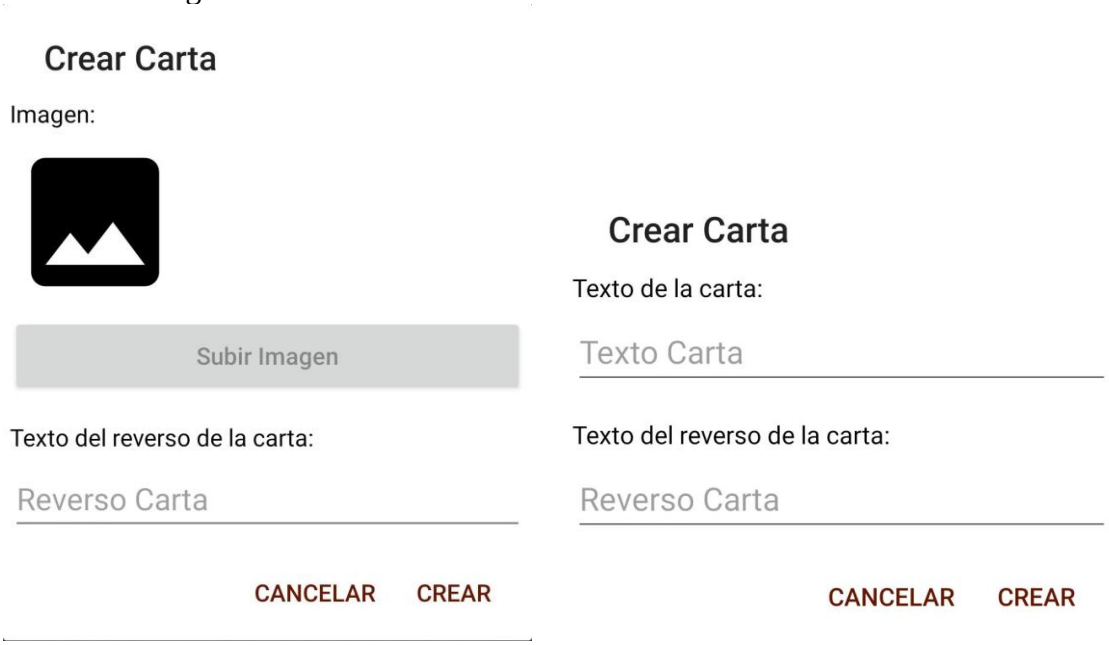

Figura 3-11: Diálogos para creación de carta

Si el usuario decide pulsar la flecha que se encuentra en la parte superior izquierda de la pantalla, será redirigido a la pantalla anterior, en este caso, la lista de mazos.

3.5.4 Pantalla de edición de tarjetas

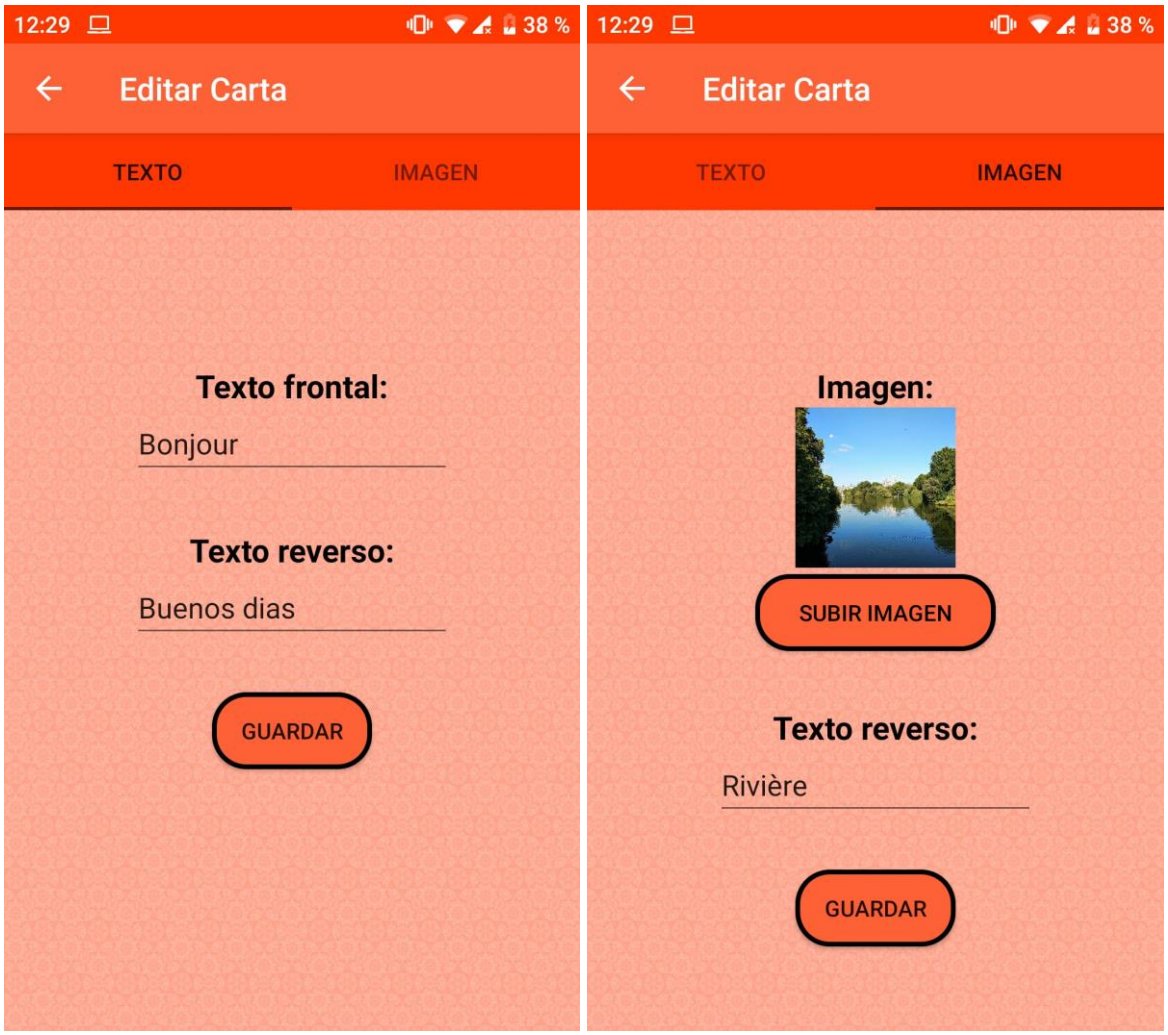

Figura 3-12: Pantalla de edición de tarjetas

Dentro de la pantalla diseñada para la edición de tarjetas, al contrario que para la creación, están unificadas la edición tanto para imágenes de solo texto como las que contienen imágenes, de modo que cualquier carta puede ser cambiada de tipo en cualquier momento tras su creación.

Para alternar entre los diferentes fragmentos, se puede pulsar en cada una de las diferentes pestañas o deslizando el dedo por la pantalla hacia cada uno de los lados, de modo que el programa hace una transición entre uno y otro.

Habiendo introducido los datos que el usuario desee para que contenga su tarjeta, tras darle al botón de guardar, la carta se actualizará. Cada botón es independiente, de modo que, si actualiza la imagen de la tarjeta, pero le da al botón de la pantalla de texto, se actualizará la tarjeta con los datos de la pestaña de texto y viceversa. Tras esto, se devolverá al usuario a la pantalla anterior.

En el caso que el usuario decida pulsar la flecha de la parte superior izquierda, será redirigido a la pantalla del listado de cartas del mazo.

3.5.5 Pantalla de resumen del mazo

Figura 3-13: Pantalla de resumen del mazo

Tras haber seleccionado uno de los mazos disponibles dentro de la lista personal del usuario, se mostrará esta pantalla.

En esta pantalla, el usuario puede ver la información referente al mazo que ha seleccionado, tanto el nombre del mazo, como la fecha del último repaso que ha realizado y las cartas disponibles para el día actual.

Toda esa información se actualizará automáticamente en el momento en el que el usuario termine cada uno de los repasos que realice sobre el mazo dentro de la aplicación.

El botón de la parte inferior le permitirá al usuario empezar el repaso del mazo con las cartas que tiene disponibles y, en el caso de que ese día no tuviera o no le quedasen más cartas que repasar, ese botón no se mostrará en la pantalla.

Pulsando la flecha de la parte superior izquierda el usuario será redirigido a la pantalla del listado de mazos general del usuario.

3.5.6 Pantalla de repaso del mazo

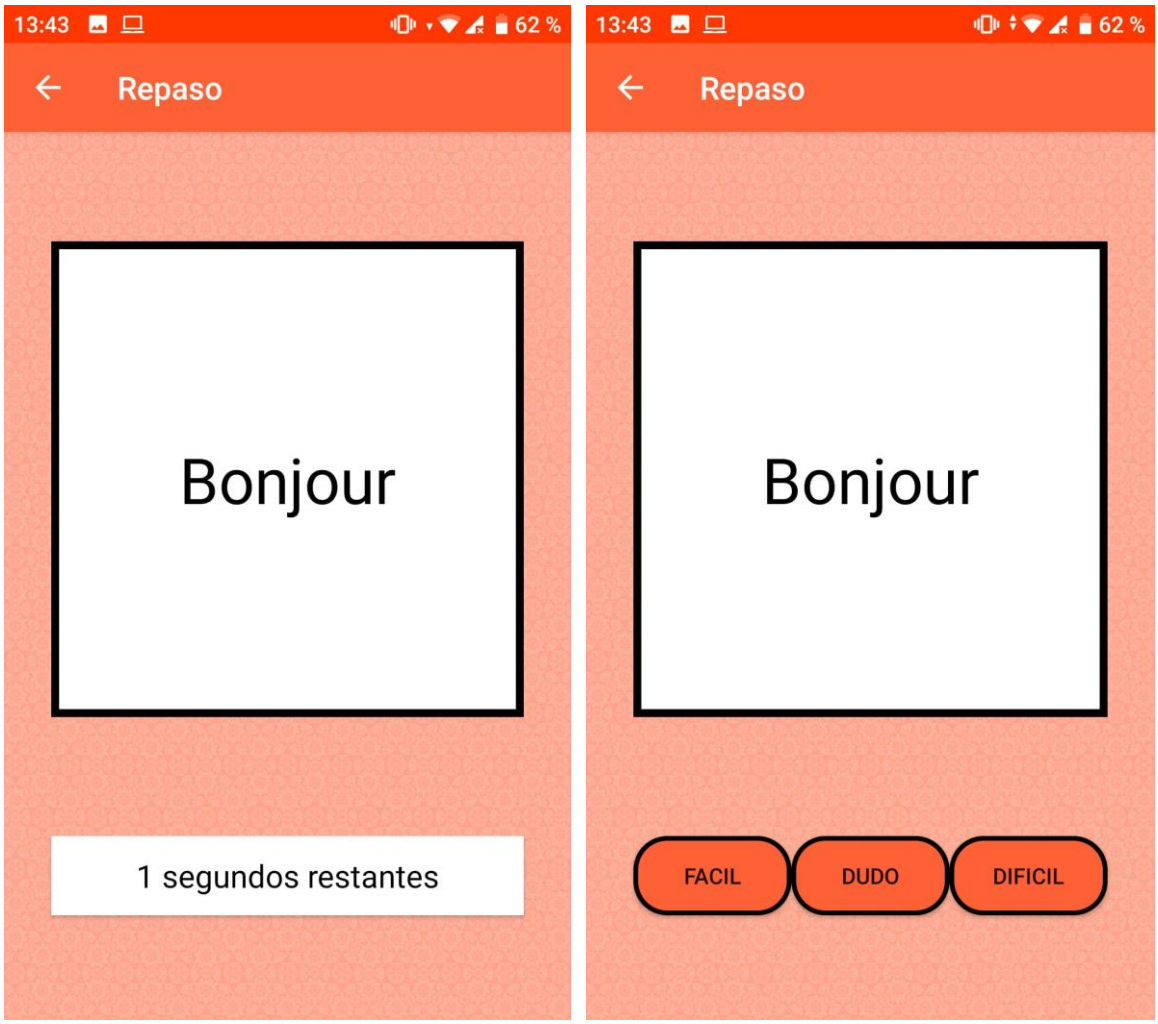

Figura 3-14: Pantalla de repaso del mazo

Esta es la pantalla en la que el usuario debería pasar más tiempo a lo largo del uso de la aplicación, ya que en ella se realizarán los repasos de las tarjetas que el usuario haya creado para su estudio y aprendizaje.

Nada más aparecer una tarjeta, el usuario podrá ver la parte frontal de ella y un recuadro en la parte inferior que le dará un tiempo antes de dejarle interactuar con el resto de los elementos, que se encuentran ocultos o inactivos. Pasados estos segundos, se mostrarán los tres botones de las opciones del algoritmo y el usuario podrá interactuar con la tarjeta pulsando sobre ella. Cada vez que pulse sobre la tarjeta, esta mostrará una animación de giro que alternará entre la parte frontal y la trasera de la tarjeta.

Cuando el usuario pulse sobre alguno de los botones de la parte inferior, se mostrará la siguiente tarjeta disponible habiendo actualizado los valores de la anterior dependiendo del botón pulsado.

En el caso de que el usuario decida que quiere tomarse un descanso del repaso, en cuanto pulse la flecha de la parte superior izquierda o, en este caso concreto, pulse el botón para volver a la pantalla anterior de su dispositivo móvil, se le mostrará el siguiente diálogo:

Figura 3-15: Diálogo de finalización del repaso

Si el usuario decide salir tras habérsele mostrado el diálogo, la aplicación le devolverá a la pantalla de repaso del mazo, mientras que actualiza los cambios de este en la base de datos de Firebase.

4 Desarrollo

En este capítulo de la memoria se va a explicar todo el proceso de desarrollo a lo largo del proyecto, desde las bases de las que parte, hasta la versión actual con todas sus funcionalidades.

La aplicación ha sido desarrollada en el lenguaje de programación Kotlin, debido a que es uno de los lenguajes más usados para el desarrollo de software para dispositivos móviles que usan el sistema operativo Android. Además, también ha sido usado el lenguaje XML, para la definición de los ficheros de recursos del programa para el diseño de las vistas y definición de variables de entorno usadas de forma global dentro del programa, tanto en los ficheros Kotlin como en el resto de los ficheros XML.

4.1 Algoritmo SuperMemo2

En esta sección se va a hablar sobre el algoritmo usado para personalizar el aprendizaje de cada usuario. Este algoritmo será usado dentro de cada uno de los repasos que haga el usuario dentro de la aplicación.

Antes de empezar a explicar en detalle en qué consiste el algoritmo SuperMemo2, cabe explicar en qué consiste el repaso espaciado para el cual está diseñado este algoritmo.

4.1.1 Repaso espaciado

El repaso espaciado [4] consiste en un conjunto de técnicas de aprendizaje con las cuales realizas ejercicios de memoria con un conjunto de datos. Estos ejercicios se van espaciando a lo largo del tiempo, incrementando el intervalo entre unos y otros según se vayan realizando.

El repaso espaciado está basado en la curva del olvido de Hermann Ebbinghaus [5], el cual decidió memorizar un gran número de palabras sin sentido y fue apuntando el número de palabras que recordaba a lo largo del tiempo y cómo iba decreciendo ese número al no ir haciendo repasos de las mismas. Al representar estos datos obtenemos la "Curva del olvido":

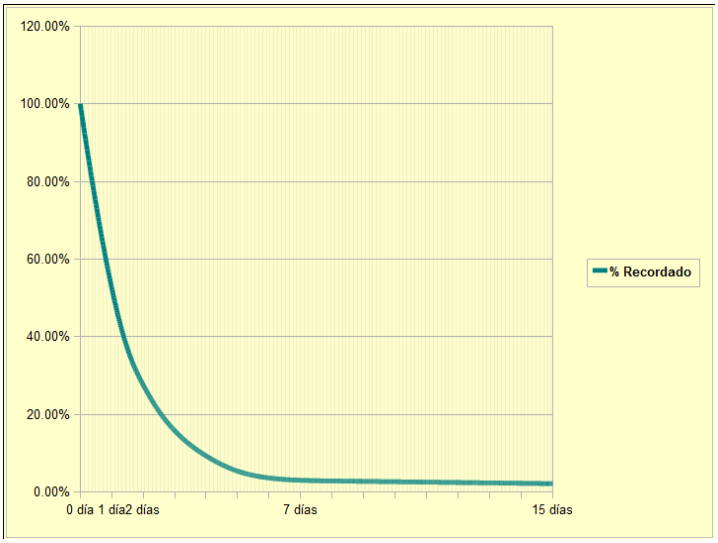

Figura 4-1: Curva del olvido

Para reducir esta progresión del olvido, el punto clave está en reducir la velocidad de éste, ya que, haciendo esto, la curva tomaría una forma menos pronunciada y con mayores valores de recuerdo pasado el tiempo.

La solución para poder reducir la velocidad del olvido es hacer repasos a lo largo del tiempo sobre lo que estamos intentando aprender. De este modo, reduciremos el olvido que tenemos sobre ello y los datos nos durarán más tiempo en la memoria.

Pero hay un detalle sobre estos repasos y es que no basta con hacerlos, sino que hay que saber cuándo hacerlos. Como nos explican desde Ncase [6], la clave de estos repasos es saber adaptarlos para encontrar nuestro punto óptimo a partir de nuestra velocidad de olvido, de modo que podamos reducir el número de repasos para mantenernos en un nivel bueno de memoria sobre lo que estamos intentando aprender.

Figura 4-2: Curva del olvido con repasos espaciados

4.1.2 Caja de Leitner

Para poder poner a prueba el repaso espaciado, Sebastian Leitner [7] diseño un sistema que implementaba el repaso espaciado. Usando tarjetas que contuvieran todo el material a aprender en forma de preguntas (por un lado, la pregunta y por otra la respuesta) y utilizando un conjunto de cajas las cuales correspondían cada una a un nivel de dificultad.

Cada día se repasaban un conjunto de cajas distinto, dado que cada caja se repasaba cada X días, siendo X el nivel que corresponde a la caja. De las cajas de cada día, se repasan en orden de mayor a menor nivel. Las tarjetas que se acierten suben un nivel y las que se fallen vuelven al nivel 1, independientemente de en qué caja se encuentren. En el caso de que la tarjeta esté en el último nivel y se acierte, se la retirará de las cajas, dándola como aprendida.

Figura 4-3: Sistema de Leitner

Con este sistema, lo que se asegura es que se refuercen más las tarjetas que más cuesta recordar, dado que las cajas de menor nivel son repasadas más frecuentemente y, por lo tanto, las tarjetas que mejor se lleven, se repasarán menos veces hasta que finalmente puedan ser sacadas del sistema de cajas.

4.1.3 SuperMemo2

El algoritmo escogido para ser utilizado por la aplicación para seguir el sistema de repaso espaciado es SuperMemo2 [8]. SuperMemo2 es uno de los algoritmos más usados por este tipo de programas para personalizar el aprendizaje a cada usuario a partir de sus interacciones dentro de los repasos ofrecidos.

En la aplicación se hará uso de una versión personalizada del algoritmo, para hacer su uso más sencillo y cómodo para el usuario, ya que, normalmente, el algoritmo le presenta al usuario seis opciones, según la dificultad que haya tenido para recordar lo que esté intentando aprender en ese momento (en el caso de esta aplicación, tarjetas) y, en el caso de la aplicación, solo se le mostrarán tres valores: difícil, dudo y fácil. Por lo tanto, si el rango de valores de SuperMemo2, según la selección del usuario, está entre 0 y 5, en SpacedCards se tomarán 1 para difícil, 3 para dudo y 5 para fácil.

El algoritmo, a partir de los valores de entrada por parte del usuario y, junto a los valores que vaya teniendo la carta para las distintas variables, va a realizar el cálculo para añadir el número de días correspondientes a la fecha actual para obtener la nueva fecha de repaso de la tarjeta. Las variables del algoritmo, también explicadas en el punto (3.4.1), son las siguientes:

- **Quality:** Entrada por parte del usuario que indica la dificultad que ha tenido para recordad la tarjeta.
- **Repetitions:** Número de veces seguidas en las que el usuario no ha podido recordar fácilmente la tarjeta.
- **Easiness:** Facilidad de la carta acumulada a lo largo de los repasos.
- **Interval:** Número de días calculados hasta el siguiente repaso.
- **NextPracticeDay:** Nuevo día de repaso calculado para la tarjeta.
- **NextPracticeMonth:** Nuevo mes de repaso calculado para la tarjeta.
- **NextPracticeYear:** Nuevo año de repaso calculado para la tarjeta.

Este algoritmo se ejecutará dentro de los repasos de los mazos por parte del usuario, en los que se le mostrarán cada una de las tarjetas que su fecha de repaso sea igual o posterior a la del día en que se realice dicho repaso.

El algoritmo sigue los siguientes pasos:

1. **Cálculo del nuevo valor para la facilidad de la tarjeta:**

```
easiness = max(1.3, easiness + 0.1 - (5.0 - quality) * (0.08 + (5.0 -
                        quality) * 0.02))
```

Código 4-1: Cálculo del nuevo valor para la facilidad de la tarjeta

Este cálculo se realiza a partir de la facilidad anterior al repaso que tenía la tarjeta y escogiendo el máximo entre dos valores distintos:

- **1.3** → Valor mínimo de facilidad que puede tener la tarjeta en el algoritmo.
- **easiness + 0.1 - (5.0 - quality) * (0.08 + (5.0 - quality) * 0.02))** → Nuevo valor calculado por el algoritmo a partir de modificar la facilidad anterior de la carta con los nuevos valores obtenidos por la entrada del usuario en el repaso de la tarjeta, tras aplicarle los factores correspondientes.

2. **Comprobación de la variable repetitions:**

```
if (quality < 3) {
            repetitions = 0
        } else {
            repetitions += 1
        }
```

Código 4-2: Comprobación de la variable repetitions

Como se ha explicado anteriormente, la variable *repetitions*, hace referencia al número de veces seguidas que al usuario le ha resultado difícil recordar la tarjeta en cuestión. Por lo tanto, cada vez que el usuario seleccione las opciones de *Dudo* y *Fácil*, el contador de repeticiones volverá a 0 y, en el caso de darle a la opción de *Difícil*, el contador aumentará en 1 su valor.

3. **Cálculo del intervalo entre repasos de la carta:**

```
if (repetitions <= 1) {
            interval = 1
        } else if (repetitions == 2) {
            interval = 6
        } else {
            interval = round(interval * easiness).toInt()
        }
```

Código 4-3: Comprobación de la variable repetitions

Una vez obtenido el número de repeticiones de la tarjeta ya se puede calcular el intervalo de días que va a pasar entre un repaso y otro de la tarjeta. Como se ha explicado en el punto (4.1.1) estos repasos tienen que ser incrementales a partir de que se vaya recordando correctamente la información de la tarjeta. Por lo tanto, el intervalo puede tener tres valores:

- **1** → Si entre los dos últimos repasos de la tarjeta ha habido al menos una vez en la que el usuario ha escogido la opción de *Difícil* al hacer el repaso de la tarjeta.
- **6** → Si tras un repaso seleccionando *Difícil* el usuario ha tenido los dos últimos repasos seleccionando una opción entre *Dudo* y *Fácil*, lo que quiere indicar que el usuario está empezando a recordar correctamente la tarjeta.
- **round(interval * easiness).toInt()** →Cuando el usuario lleva tres o más repasos seguidos en los que no ha escogido nunca la opción de *Difícil* el algoritmo calcula el nuevo intervalo a partir del intervalo anterior que tuvo la tarjeta y el valor calculado en el paso 1 para la facilidad de la carta. De este modo se asegura que los intervalos vayan siendo crecientes ya que, el valor de *easiness* va a ser siempre mayor o igual que 1.3. El valor obtenido es redondeado con la función *round* para eliminar su parte decimal y pasado a número entero con la función *toInt* para su uso en el paso posterior.

4. **Cálculo de la nueva fecha de repaso:**

```
val date = Calendar.getInstance()
date.add(Calendar.DATE, interval)
        nextPracticeDay = date.get(Calendar.DAY_OF_MONTH)
        nextPracticeMonth = date.get(Calendar.MONTH) + 1
        nextPracticeYear = date.get(Calendar.YEAR)
```

Código 4-4: Cálculo de la nueva fecha de repaso

Por último, a partir de los valores obtenidos a lo largo del algoritmo, se calcula la nueva fecha de repaso de la tarjeta. Para ello, lo primero que se hace es obtener la fecha actual a partir de una instancia de la librería Calendar, proporcionada por la librería *java.util*. Esta librería, además de proporcionarnos la fecha del instante en el que solicite, nos proporciona funciones para el manejo de estas fechas. En este caso se hace uso de la función *add*, la cual nos permite añadir un número entero para calcular la nueva fecha con sus cambios de mes y año correspondientes, si se diera el caso. Por último, haciendo uso de la función *get*, también proporcionada por la librería, podemos sacar los valores de día, mes y año de la fecha para su posterior almacenamiento en la base de datos como enteros separados. Como se explicó en el punto (3.4.1), esto es necesario, ya que el trabajo a partir de ficheros JSON para el intercambio de datos dentro del programa no nos permite trabajar de forma sencilla con estructuras complejas, como es el caso de la clase Calendar, y es necesario dividir la fecha en distintas variables sencillas y luego volver a reunirlas cuando sea necesario.

4.2 Desarrollo de los sistemas de la aplicación

En esta sección se va a explicar cómo se ha seguido y en que ha consistido la parte de código del desarrollo de la aplicación. Primero de todo, se va a dar un marco general de cosas en común que tiene la aplicación y luego se va a ver algo más detallado por cada sistema de los descritos en el apartado (3.1).

Todo el código de la aplicación se ha desarrollado dentro del entorno de Android Studio, proporcionado por el propio Android y de acceso gratuito para todos los usuarios.

Dentro del proyecto creado dentro de este entorno, los ficheros fuente se encuentran en dos grandes carpetas dentro del proyecto:

- **Paquete java.com.tfg.spacedcards →** Donde se encuentran todos los ficheros de código que desempeñan las funciones que va a realizar la aplicación.
- **Carpeta res →** Donde se encuentran todos los ficheros XML de estilo y diseño.

Dentro del paquete principal del proyecto nos podemos encontrar las siguientes carpetas:

- **Activities →** En su interior se encuentran todos los ficheros que definen las actividades de las vistas de la aplicación, algunos de los fragmentos usados y los diálogos, así como algún fichero necesario para hacer funcionar algunos elementos del diseño correctamente.
- **FB →** Dentro se encuentra el fichero necesario para la conexión con el servidor de Firebase (***FBDataBase.kt***)
- **Login →** Contiene los ficheros necesarios para el funcionamiento del login. Son una versión adaptada para el proyecto a partir de los generados por el propio Android Studio.
- **Model →** Agrupa los ficheros que contienen las clases con su modelo de datos respectivo para ser usadas en el resto de la aplicación (Mazos, Tarjetas, Usuarios…)

4.2.1 Sistema de Usuarios

El sistema de usuarios es el encargado de controlar el inicio de sesión, el registro y la solicitud de reinicio de contraseña de los usuarios.

Dentro de la carpeta *Login* se encuentra el fichero *LoginActivity.kt*, el cual es el encargado de controlar la vista del acceso del usuario a la aplicación.

En la creación de la vista se recogen todos los elementos del diseño (botones, campos de texto…) para poder ser usados como variables en el código y poder recoger sus valores e interactuar con ellos.

En este fichero se hace uso de dos elementos importantes a destacar:

- **Observers →** Dentro de la vista estamos definiendo los campos del formulario como observers. Esto nos permite que tanto el campo de introducción del email del usuario como el de su contraseña lancen un evento cada vez que su valor se actualice, es decir, cuando el usuario introduzca o elimine algún carácter dentro de

ellos. En este caso, es usado para el control de los textos introducidos por el usuario, dado que el email y la contraseña deben cumplir unos requisitos mínimos para poder continuar con el proceso. Los botones de inicio de sesión y de registrase estarán deshabilitados hasta que el usuario introduzca un correo con un formato válido y una contraseña con una longitud superior a 5 caracteres.

- **Listeners** → Los listeners son elementos a los que se les asigna una acción a realizar a partir de un evento. Estos eventos son de un tipo concreto (con el que se haya interactuado con el elemento que tiene asociado el listener). En el caso de la aplicación, los botones y el texto para cambiar contraseña siguen este diseño, tienen asignado un listener para cuando el usuario pulsa sobre ellos, en dicho caso, se lanzará la acción pertinente.

Todas las acciones finales realizadas desde la vista de acceso (inicio de sesión, registro o mostrar diálogo de petición de reinicio de contraseña) acaban realizando una llamada a una función de las que implementa el fichero *FBDataBase.kt*. Dichas funciones, en el caso de las llamadas de este sistema, todas invocan una función de la librería de apoyo de Firebase para el sistema de autenticación, usando los parámetros recogidos de los formularios de la vista.

4.2.2 Sistema de Mazos

El sistema de mazos es el encargado de mostrar y listar los mazos además de permitirle al usuario la creación y gestión de estos.

Para el listado de los mazos se ha hecho uso de una vista recicladora (*RecyclerView*) para mostrarlos dentro de la aplicación. Este tipo de vistas son de las más usadas por los desarrolladores cuando se tiene una pantalla en la que se lista una gran cantidad de datos y es necesario usar desplazamientos en ellas para poder ver todas las entradas. Para poder utilizarla dentro de la aplicación es necesario hacer uso del widget, proporcionado por la librería de *androidx*, en el fichero de diseño.

Dentro de la vista recicladora, cada una de las entradas corresponde a un adaptador el cual dentro contiene un único mazo y diferente al del resto de adaptadores. Este adaptador puede personalizarse para que muestre la información del mazo que se desee y con el diseño que se prefiera, dado que posee su propio fichero de diseño. Para poder usar los adaptadores personalizados para la aplicación se creó un fichero *Extensions.kt*, el cual es usado por toda la aplicación para diferentes funcionalidades que se querían añadir a clases proporcionadas por librerías que no tenían toda la necesaria para el desarrollo del proyecto.

Para integrar la vista recicladora dentro de la vista se hizo uso de los fragmentos, que nos permiten representar el comportamiento de una parte de la interfaz en vez de toda en su totalidad, como pasa con las actividades. Estos fragmentos son incluidos dentro de una actividad y, para el caso del listado de los mazos, nos permite actualizarlo sin tener que actualizar la actividad entera, de modo que el funcionamiento de la aplicación es más fluido y rápido que incluyendo la lista de forma directa en la actividad. Además, en cada momento que se cree o se actualice alguno de los mazos del usuario, este listado se actualizará al momento.

Para obtener la lista de los mazos, en cada actualización del listado, el sistema hace una llamada a la función añadida a la clase *RecyclerView* dentro del fichero de extensiones para

usar la actualización usando los adaptadores personalizados que se han creado. Dentro de esta función se realiza una llamada a la función *getCards* del fichero *FBDataBase.kt:*

```
var decks = listOf<Deck>()
val firebaseAuth = FirebaseAuth.getInstance()
for(postSnapshot in dataSnapshot.child(firebaseAuth.currentUser!!.uid).child(
"decks").children) {
    val deck = postSnapshot.getValue(Deck::class.java)!!
    decks += deck
}
```

Código 4-5: Obtención de los mazos de un usuario

La función utiliza las dos bases de datos de la aplicación, tanto la de usuarios como la de datos. Esto es necesario ya que con la instancia de la base de datos de autenticación se puede sacar el id del usuario con la sesión iniciada en el dispositivo actualmente y, con ese id, se puede acceder a su entrada del fichero JSON de la otra base de datos para obtener sus mazos. Luego, dentro del bucle, el programa va añadiendo al listado de mazos que se va a devolver todos los mazos del usuario, sacándolos ya como variables de la clase *Deck* definida en *Models.*

Para poder crear un mazo nuevo, se ha de pulsar sobre el botón flotante de la parte inferior derecha de la vista. El listener que tiene asociado lanzará el código para generar un nuevo diálogo que le permita al usuario introducir el nombre del mazo y, tras aceptar, hacer las llamadas a la función que añadirá el mazo dentro del usuario en Firebase. Para que se actualice correctamente la base de datos y se añada el mazo a la entrada del usuario el proceso que realiza el sistema es el siguiente:

- El sistema define una función callback que se va a pasar a la función que llama a Firebase junto al nombre del mazo.
- La función le añade un listener al usuario actual para controlar la inserción del nuevo mazo.
- Cuanto salta el listener, el sistema obtiene el usuario actual asignado en una variable del tipo *User* definido en la carpeta *Models.*
- Añade el mazo nuevo al usuario auxiliar que se ha definido y, en caso de no haber problemas, actualiza el valor del usuario entero en la base de datos.

En caso de que el usuario decida eliminar alguno de sus mazos, tras pulsar sobre el icono de la "X" definida a la derecha de su adaptador, se invocará un nuevo diálogo el cual servirá de confirmación por parte del usuario para eliminar de la base de datos los datos asociados a ese mazo.

Para el requisito de repaso de los mazos, el usuario ha de pulsar en el nombre del mazo que desee y se le pasará a una vista con el resumen del mazo. Siempre que haya que mandar un mazo de una vista a otra, el sistema lo hace a partir del uso de estructuras JSON, dado que simplemente con la definición de dos funciones sencillas dentro de la clase modelo de los mazos, estas estructuras se pueden tratar fácilmente para el paso de la información.

Dentro de la pantalla el sistema simplemente irá recogiendo la información del mazo recibido vía JSON y le mostrará la más importante al usuario. Para mostrar las tarjetas

disponibles, el sistema recorre el conjunto de tarjetas del mazo y mediante una función del modelo de las tarjetas se puede comparar si están en fecha para ser repasadas o no, haciendo uso de la librería *Calendar*. Si no estuvieran en fecha ninguna de las tarjetas del mazo, el sistema le oculta el botón al usuario para evitar posibles errores.

Si el usuario ha decidido empezar un nuevo repaso, se le mandará a la nueva vista el mazo entero para su repaso. En este repaso, al usuario se le van mostrando una a una las tarjetas que están en fecha y, tras pulsar alguno de los botones que se encuentran en la parte inferior, el sistema comenzará a aplicarle a la tarjeta el algoritmo explicado en el capítulo (4.1.3) y actualizará su valor dentro del mazo.

El repaso comienza con una cuenta atrás establecida de tres segundos tras mostrarse la parte frontal de la tarjeta, en la que los botones y el volteo de la tarjeta estarán desactivados hasta su finalización. Una vez terminada, el usuario podrá interactuar con dichos elementos. La tarjeta posee una animación de volteo ofrecida por la librería *EasyFlipView* de Wajahat Karim [9] la cual, mediante su integración y configuración con un par de funciones, nos permite añadirle esta funcionalidad a la vista para darle más dinamismo.

El repaso puede terminar de alguna de las siguientes dos formas:

- Ya no queda ninguna tarjeta para el día del repaso.
- El usuario ha pulsado el botón de volver de su dispositivo o la flecha de la parte superior izquierda de la vista, donde se les mostrará un diálogo para confirmar que desean terminar en ese punto el repaso y guardar su progreso.

  Para controlar que el usuario ha pulsado el botón de volver de su dispositivo, Android nos ofrece la función *onBackPressed* la cual se puede sobrescribir para que cuando salte el evento, se ejecute el código que nosotros deseemos, en este caso lanzar el diálogo:

```
override fun onBackPressed() {
       val dialog = EndTryDialogFragment(this::updateUsedDeck)
       dialog.show(supportFragmentManager, "ENDTRY_DIALOG")
   }
```

Código 4-6: Función para controlar el evento del botón de volver del dispositivo

En el momento en el que se termina el repaso, se invocan las llamadas a las funciones del fichero *FBDataBase.kt*, que actualizarán los mazos del mismo modo que en su creación, pero simplemente actualizando la entrada del mazo que corresponda.

4.2.3 Sistema de Tarjetas

El sistema de tarjetas es el encargado de permitirle al usuario la creación y gestión de las tarjetas de sus mazos, así como la subida de imágenes desde el almacenamiento del dispositivo a la aplicación.

Al igual que con los mazos del punto anterior (4.2.2), para listar las tarjetas se hace uso de los fragmentos para organizar el contenido dentro de una vista, dividiéndolo en diferentes secciones y pudiendo controlar cada una por separado. Se hace uso de las vistas recicladoras proporcionadas por las librerías por *androidx* para poder listar las tarjetas y que se muestren con sus propios adaptadores personalizados, los cuales, en este caso, están

diseñados para mostrar la información de cada lado de la tarjeta, mostrando la imagen en el caso que sea necesario.

En la obtención del listado de tarjetas de cada uno de los mazos se hace uso del mismo método usado para los mazos. La vista recicladora se actualiza con la función extendida de la clase dentro del fichero *Extensions.kt*, pero esta vez usando el adaptador creado para las tarjetas. Esta función hará uso de la función diseñada en el fichero *FBDataBase.kt*, la cual accede a la entrada del mazo dentro del usuario con la sesión iniciada en el JSON de la base de datos de la aplicación y devuelve el listado con las tarjetas que haya dentro de él.

Para poder crear una nueva tarjeta, el usuario deberá pulsar en el botón flotante de la parte inferior derecha de la vista de edición de los mazos. Este botón lleva asociado un listener, el cual, tras ser activado, muestra una animación donde se despliegan (o se repliegan) los dos botones secundarios que le permitirán al usuario la creación de los diferentes tipos de tarjetas. Esta animación funciona de la siguiente manera:

- **Desplegar** → Tras pulsar sobre el botón principal, los dos botones secundarios son desplazados en el eje vertical hasta situarse en el punto deseado. Tras esto, se le asigna a cada uno su respectivo listener para la creación de cada tipo de tarjeta.
- **Replegar** → Tras pulsar sobre el botón principal, los dos botones secundarios son desplazados hasta situarse detrás del botón principal siendo totalmente tapados por este. Tras esto, se le quita a cada uno el listener que tenían asociado poniendo su valor a *null*.

Tras pulsar sobre cualquiera de los dos botones, saltará el diálogo que le permita al usuario introducir los datos para crear sus tarjetas propias. En el caso de las tarjetas que contengan una imagen, el botón para subirlas desde la galería, la primera vez, le solicitará al usuario los permisos para poder acceder al almacenamiento del dispositivo. En caso de que el usuario deniegue los permisos a la aplicación, el diálogo se le seguirá mostrando hasta que acepte los permisos, impidiéndole a la aplicación el poder subir imágenes.

Una vez aceptados los permisos, el usuario accederá a la galería y podrá seleccionar una imagen mientras el programa queda a la espera del resultado proporcionado por el usuario. Tras seleccionar el archivo deseado y volver a la aplicación, se realizará una transformación de la imagen necesaria para reducir su tamaño y para poder pasarla a una cadena codificada en Base64, para poder almacenarla y pasarla a través de los JSON.

```
image_view.buildDrawingCache()
var bitmap = image_view.drawingCache
val height = bitmap.height / (bitmap.height / 300)
val width = bitmap.width / (bitmap.width / 300)

bitmap = Bitmap.createScaledBitmap(image_view.drawingCache, width, height
                                   , true)
val baos = ByteArrayOutputStream()
bitmap.compress(Bitmap.CompressFormat.JPEG, 100, baos)
val imageBytes = baos.toByteArray()
imageB64 = Base64.encodeToString(imageBytes, Base64.DEFAULT)
```

Código 4-7: Codificación de la imagen

Dentro de la vista de edición y en los repasos de los mazos, se utiliza el siguiente código para decodificar la cadena y mostrar la imagen:

```
val imageBytes = Base64.decode(deck.cards[index].imageB64, Base64.DEFAULT)
val decodedImage = BitmapFactory.decodeByteArray(imageBytes, 0,
                                                 imageBytes.size)
front.setImageBitmap(decodedImage)
```

Código 4-8: Decodificación de la imagen

Para la edición de las tarjetas se ha diseñado una vista por pestañas la cual contiene diferentes fragmentos para su implementación:

- **FragmentPagerAdapter** → Dentro de este fragmento definimos el sistema de pestañas que va a usar la aplicación y en él se incluyen las instancias a los fragmentos para edición de las tarjetas.
- **EditTextCardFragment** → Fragmento para poder editar las tarjetas con solo texto en ambas caras.
- **EditImageCardFragment** → Fragmento para poder editar las tarjetas con imagen en una cara y texto en otra.

Cuando se pulsa sobre el botón de editar una tarjeta, al usuario se le muestra la vista con ambas pestañas, independientemente del tipo de carta que sea, así se le permite cambiar el tipo de carta en todo momento. Este sistema, además de comodidad, le permite al usuario interactuar deslizando horizontalmente la pantalla para navegar entre una pestaña y otra, además de pulsando sobre la propia pestaña. Finalmente, cuando pulse sobre el botón de guardar, si los campos no son vacíos, el sistema guardará la tarjeta con los datos y el tipo de la pestaña en la que se encuentre el usuario, haciendo uso de las funciones de *FBDataBase.kt* para su almacenamiento en la base de datos del servidor.

5 Integración, pruebas y resultados

En este capítulo de la memoria se va a mostrar cómo se ha realizado la integración de la aplicación en los dispositivos de los usuarios y algunas de las pruebas que se han realizado sobre la aplicación.

Como la aplicación se ha desarrollado exclusivamente para dispositivos móviles que utilizan el sistema operativo Android, la aplicación es instalada mediante un fichero APK.

Las primeras pruebas que se realizaron fueron para recoger la experiencia de usuario, probando la aplicación en un grupo de distintos dispositivos para ver el rendimiento y funcionamiento en todos ellos.

Estas pruebas sirvieron para poder adaptar mejor diferentes elementos de las vistas que o no terminaban de adaptarse bien al espacio que tenían diseñado para ellos o para detectar fallos de memoria, como pasaba a la hora de subir las imágenes, y así mejorar el funcionamiento del tratamiento de estas por parte de la aplicación.

Una vez pasadas estas pruebas, a partir de una de las herramientas que ofrece Android Studio, se puede estudiar el rendimiento de la aplicación en términos de CPU, Memoria e Internet:

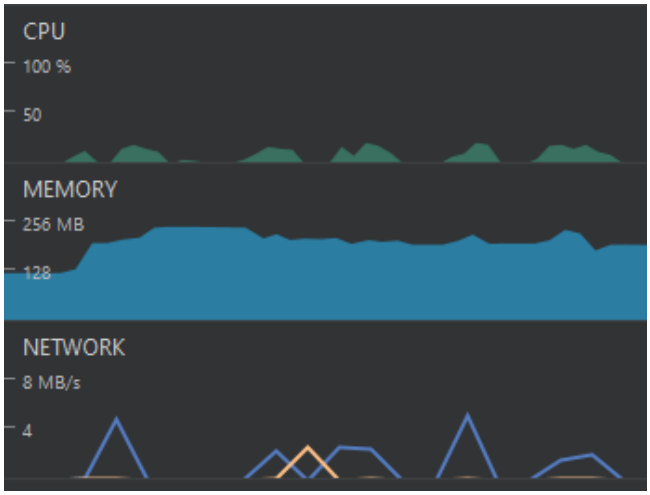

Figura 5-1: Rendimiento de la aplicación

La figura representa el comportamiento de la aplicación durante la subida a la base de datos de una nueva tarjeta con imagen, además de la posterior actualización de la vista recicladora tanto de las tarjetas, como de la lista de los mazos del usuario. Todo este escenario cuenta con el usuario teniendo ya creados más mazos, aparte del modificado, y más tarjetas dentro del mismo mazo.

Como se puede apreciar, los valores son números muy aceptables para el trabajo que está desempeñando, ya que el uso de CPU no sobrepasa el 25%, el uso de la memoria está con máximos en torno a los 256MB y en cuanto a los datos de red, vemos que el consumo de datos se ha podido controlar bastante bien gracias a los tratamientos de las imágenes previamente explicados en esta memoria.

6 Conclusiones y trabajo futuro

En este capítulo de la memoria se van a dar unas conclusiones basadas en el trabajo realizado y la experiencia obtenida a lo largo del desarrollo del proyecto, así como ciertos aspectos en los que trabajar en el futuro para la mejora de la aplicación creada.

6.1 Conclusiones

La idea de partida del proyecto fue la de crear una aplicación sencilla y cómoda para que el usuario, en este caso un profesor, pudiera tener una herramienta de refuerzo para poder aprenderse los nombres de sus alumnos y así ofrecerles una mejor asistencia a la hora de sus explicaciones y ayudas durante las clases.

A lo largo del desarrollo se han ido incrementando esas funcionalidades permitiendo que este objetivo inicial sea solo uno de los muchos de usuarios potenciales para los que actualmente se encuentra diseñada la aplicación.

Se ha logrado crear una aplicación que, mediante una interfaz sencilla, pueda implementar una de las técnicas más conocidas para el refuerzo del aprendizaje, usando uno de los mejores algoritmos para desarrollar esta práctica. Además, la aplicación ha demostrado tener buenos números en cuanto a rendimiento, los cuales le convierten en una aplicación muy útil y viable para usarla como herramienta de apoyo en todo proceso de aprendizaje que se desee.

6.2 Trabajo futuro

Dado que es una aplicación que va a estar en contacto directo con el usuario, siempre existirán aspectos visuales que puedan ser mejorables para facilitar y mejorar la experiencia del mismo durante el uso de la aplicación.

Existen otros aspectos en los que se puede mejorar la funcionalidad de la aplicación para ofrecerles un abanico de oportunidades mayores a los usuarios que hagan uso de ella. Entre ellos destaca la implementación de un sistema de almacenamiento de mazos compartidos a los que los usuarios puedan subir sus propios mazos y poder suscribirse y descargarse mazos que hayan diseñado y subido otros usuarios, para así facilitar el proceso de creación de estos y poder crear una comunidad en torno a la aplicación a través de este sistema.

También, otro de los aspectos para tener en cuenta, sería el desarrollo para otros sistemas, como sería el caso de los dispositivos que hacen uso del sistema iOS, dando así una ampliación de posibles usuarios que hagan uso de la aplicación.

Referencias

[1] 42matters (2019, Dec 29). Store Stats for Mobile Apps [Online]. Available: https://42matters.com/stats

[2] AppBrain (2019, Dec 28), Top Categories on Google Play [Online]. Available: https://www.appbrain.com/stats/android-market-app-categories

[3] Statcounter (2019, Dec). Mobile Operating System Market Share Worldwide [Online]. Available: https://gs.statcounter.com/os-market-share/mobile/worldwide/#monthly-200901-201912

[4] D. Rohrer and K. Taylor, "The effects of overlearning and distributed practise on the retention of mathematics knowledge.", Applied Cognitive Psychology: The Official Journal of the Society for Applied Research in Memory and Cognition, 20(9), 1209-1224, 2006.

[5] Luis (2010, Jan 21). La curva del olvido (Ebbinghaus) ¿Cuánto recuerdas si no lo repasas? [Online]. Available: https://www.elartedelamemoria.org/2010/01/21/curva-del-olvido-ebbinghaus/

[6] Nicky Case (2018, Oct). How to remember anything forever-ish [Online]. Available: https://ncase.me/remember/es.html

[7] Mnemotecnia (2016, Dec 3). Sistema Leitner [Online]. Available: https://www.mnemotecnia.es/la-pastilla-verde-sistema-leitner

[8] P. Wozniak, "Optimization of learning", Master's Thesis, University of Technology in Pozna, 1998.

[9] Wajahat Karim (2018, Dec 12). EasyFlipViewPager – The Flip Animations For Your ViewPager [Online]. Available: https://android.jlelse.eu/easyflipviewpager-the-flip-animations-for-your-viewpager-fd66b34f4703

Glosario

Android	Sistema operativo para dispositivos móviles
Android Studio	Entorno de desarrollo para aplicaciones Android
API	Application Programming Interface
APK	Fichero de instalación de aplicaciones Android
Firebase	Plataforma de desarrollo proporcionada por Google para aplicaciones Android. Ofrece herramientas para bases de datos y gestión de usuarios en la nube.
Kotlin	Lenguaje de programación usado en el desarrollo de aplicaciones Android
Listener	Receptor de eventos generados a partir de la interacción del usuario con la aplicación. Ejecutan el bloque de código que tienen asociado tras su invocación.
Mazos	Conjunto de tarjetas de entrenamiento usados en la aplicación
Observer	Interfaz implementable cuando se quiere que una clase quiera ser notificada de cambios en otros elementos observables de la aplicación.
SuperMemo2	Algoritmo de aprendizaje basado en el repaso espaciado
Tarjetas	Objeto principal de aprendizaje de la aplicación. Consta de dos caras en las que se pueden introducir texto o imágenes.

Anexos

A Manual del usuario

Dentro de este anexo se va a proporcionar un manual para guiar al usuario dentro de la navegación y funcionalidad de la aplicación.

Nada más arrancar la aplicación se mostrará la vista de inicio de sesión. Dentro de esta vista el usuario va a poder tanto iniciar sesión introduciendo sus credenciales como poder registrase dentro de los servidores del programa.

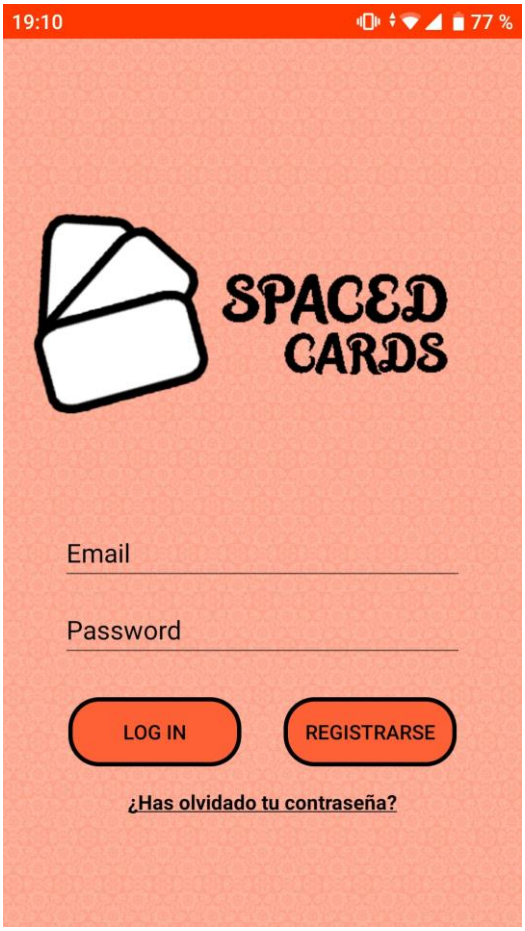

Figura A-1: Pantalla de login

El usuario deberá introducir un correo electrónico válido al igual que una contraseña que cumpla los requisitos mínimos especificados por la aplicación. El programa notificará al usuario en todo momento en caso de que estos campos no cumplan las reglas establecidas:

- Un correo electrónico del formato *usuario@dominio.com*
- Una contraseña que contenga al menos 5 caracteres

Una vez cumplidos los requisitos, el programa habilitará los botones de inicio de sesión y de registro, permitiéndole al usuario acceder a la aplicación.

Esta vista también le proporciona al usuario la posibilidad de solicitar un cambio de contraseña pulsando sobre el texto "¿Has olvidado tu contraseña?". Tras realizar esta acción, el programa mostrará un diálogo dentro del cual se debe introducir el correo con el que el usuario se registró en la aplicación. Una vez introducido, tras pulsar el botón de "Enviar", el sistema realizará las peticiones correspondientes para enviar el correo con el enlace para poder realizar el cambio de contraseña.

Figura A-2: Diálogo de cambio de contraseña

Una vez realizado el inicio de sesión, al usuario se le enseñará la vista principal de la aplicación. Dentro de ella, el usuario podrá acceder, crear y gestionar todos los mazos que haya ido diseñando a lo largo del uso del programa.

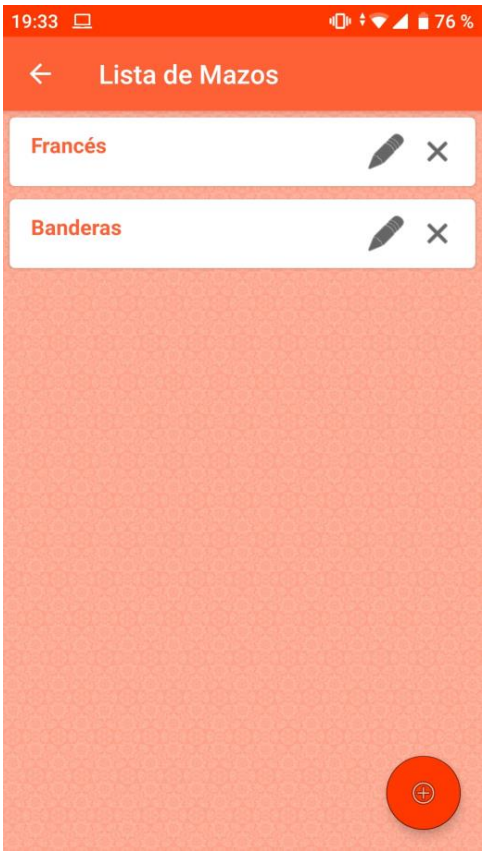

Figura A-3: Pantalla de listado de mazos

Para poder crear un nuevo mazo, el usuario ha de pulsar sobre el botón flotante situado en la parte inferior derecha de la pantalla. Tras ello, la aplicación mostrará un nuevo diálogo en el cual se le solicitará el nombre para el nuevo mazo que desea crear. Una vez se haya introducido el nombre y aceptado, el sistema comprobará que no existe ningún otro con el mismo nombre y, en caso de no existir conflictos, creará el mazo y lo mostrará sobre la lista de la vista.

Figura A-4: Diálogo de creación de mazos

Cada elemento dentro de la vista pertenece a un único mazo, de modo que todas las siguientes acciones que se realicen sobre un elemento afectarán al mazo que tiene asociado:

- Pulsando sobre el nombre del mazo, el usuario será redirigido a la pantalla de resumen del mismo, donde podrá empezar un nuevo repaso sobre él.
- Pulsando sobre el símbolo del lápiz, se mostrará la pantalla de edición del mazo.
- Pulsando sobre la “X”, aparecerá un nuevo diálogo donde se pedirá la confirmación del usuario para eliminar el mazo.

En caso de que el usuario pulse sobre la flecha de la parte superior izquierda de la pantalla, será llevado de nuevo a la pantalla de inicio de sesión, habiendo cerrado la sesión activa previamente.

Dentro de la pantalla de edición de mazos, el usuario verá mostrada una lista del mismo estilo que la que tenía en la vista principal de los mazos, pero esta vez, con las tarjetas que componen el mazo que ha decidido editar.

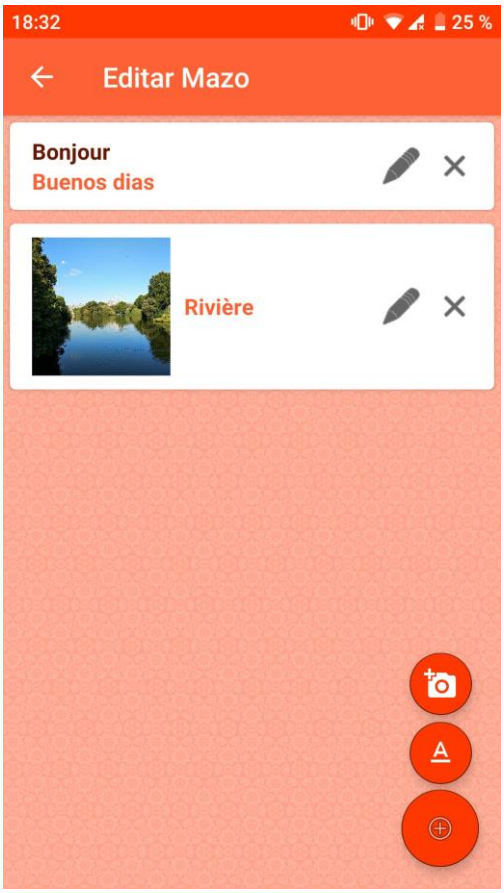

Figura A-5: Pantalla de edición del mazo

Pulsando sobre el botón de la parte inferior derecha podrá desplegar o replegar los dos botones que le permitirán crear nuevas tarjetas para su mazo:

- Pulsando sobre el botón de la cámara, se le mostrará un diálogo al usuario donde podrá diseñar una nueva tarjeta la cual contenga una imagen en su parte posterior. Para añadir la imagen, tras pulsar sobre el botón de "Subir", y haber aceptado los permisos solicitados, el programa le llevará hasta su galería. Tras haber seleccionado la imagen deseada, la aplicación recogerá la imagen y, rellenando el campo del texto trasero, podrá finalizar la creación de la tarjeta añadiéndola al mazo y actualizando este en Firebase.
- Pulsando sobre el botón con la "A", se le mostrará al usuario un diálogo donde podrá crear una tarjeta que solo contenga texto. Tras rellenar los dos campos solicitados, en el momento en el que le dé a enviar, la tarjeta se añadirá al mazo y este se actualizará dentro de Firebase.

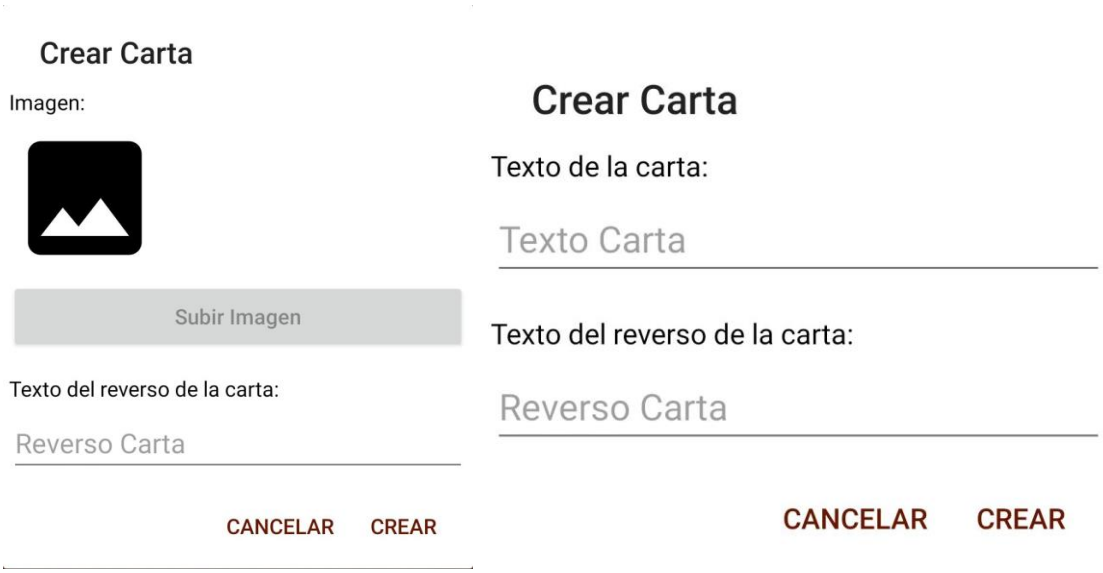

Figura A-6: Diálogos de creación de tarjetas

Si el usuario decide interactuar con el adaptador que contiene cada tarjeta, podrá realizar dos acciones distintas:

- Pulsando sobre el botón del lápiz, el sistema le mostrará al usuario la vista donde podrá editar la tarjeta en cuestión.
- Pulsando sobre la "X", la aplicación le mostrará al usuario un diálogo de confirmación para poder procedes a la eliminación de la tarjeta.

Pulsando sobre la flecha de la parte superior de la pantalla, el usuario será llevado de vuelta a la pantalla del listado de mazos.

En la pantalla de edición de tarjetas, el usuario se encontrará con una vista dividida en pestañas. Para poder cambiar entre una y otra bastará con pulsar cobre el nombre de cada una de ellas o deslizar el dedo horizontalmente dentro de las pestañas.

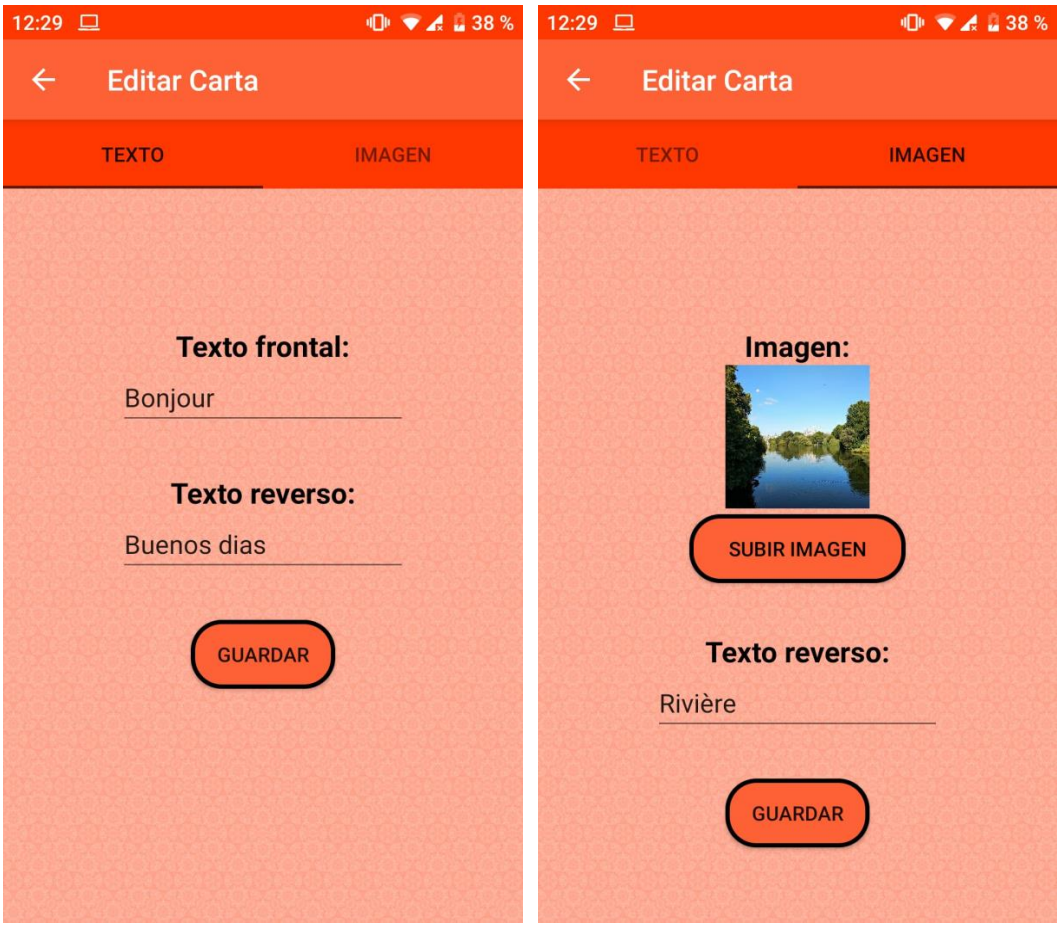

Figura A-7: Pantalla de edición de tarjetas

Cada una de las pestañas hace referencia a un tipo de tarjeta distinto. Ambas pestañas estarán disponibles para cada tarjeta, permitiéndole al usuario así poder cambiar el tipo de esta cuando desee.

Dentro de la pestaña de texto se le pedirá al usuario ambos campos de texto (frontal y reverso) con los que desea actualizar la información de la tarjeta. Dentro de la pestaña de imagen, el usuario contará con un botón para poder acceder a la galería y así subir el archivo que desee usar para colocar en el frontal de su tarjeta.

Ambos botones de guardar son independientes el uno del otro, de modo que, dependiendo de la pestaña en la que nos encontremos, la información de la tarjeta se actualizará usando los campos rellenado en la de texto o en la de imagen, dependiendo de que botón se haya pulsado.

Si el usuario pulsa sobre la flecha de la parte superior de la pantalla, será devuelto a la vista del listado de tarjetas del mazo.

Dentro de la vista de resumen del mazo el usuario podrá encontrar toda la información referente al mazo que haya seleccionado previamente desde el listado.

Figura A-8: Pantalla de resumen del mazo

Esta información disponible consta de:

- Nombre del mazo seleccionado
- Fecha del último repaso realizado sobre el mazo
- Número de tarjetas disponibles para el día actual

Toda esta información se actualiza en cada momento que se recargue esta pestaña, de modo que tras volver de los repasos que se realicen esta pantalla ya contará con la información nueva.

El botón de la parte inferior le redirige al usuario a la pantalla para realizar el repaso del mazo. Este elemento no se encontrará visible en todo momento dado que depende del número de tarjetas disponibles para cada día. En el caso en el que haya una o más tarjetas este botón estará funcional y, cuando sea cero, no aparecerá en la interfaz.

Si el usuario decide pulsar la flecha de la parte superior de la vista, será devuelto a la pantalla del listado de mazos.

Dentro de la vista del repaso del mazo, el usuario irá viendo una a una todas las tarjetas que tenga disponibles para cada día y podrá repasarlas hasta que termine el repaso o decida cortarlo en algún punto del mismo.

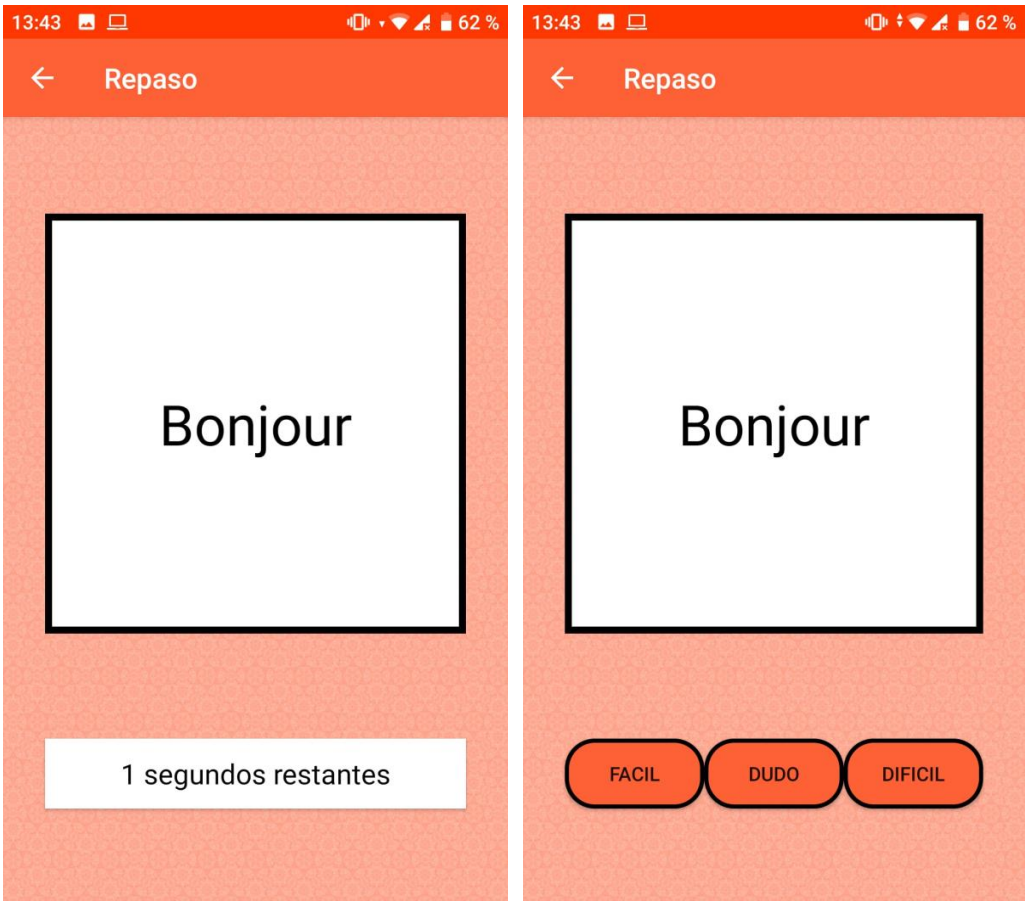

Figura A-9: Pantalla de resumen del mazo

Nada más mostrar cada tarjeta, el usuario verá la parte frontal de la misma y una cuanta atrás especificada por el sistema para darle tiempo a recordar la información que pusiera en la parte del reverso de la tarjeta. Una vez terminada la cuenta atrás, el usuario podrá interactuar con el resto de los elementos de la vista.

Si el usuario pulsa sobre la carta (habiendo terminado la cuenta atrás), esta realizará una animación de giro con la que se mostrará lo que hay en el lado no visible de la tarjeta en ese momento.

Para continuar con el repaso, el usuario ha de pulsar sobre cualquiera de los botones de la parte inferior de la pantalla. Estos botones indican la dificultad que ha tenido a la hora de recordar la información de la tarjeta, siguiendo los valores especificados de cada botón. Una vez pulsado cualquiera de ellos, la vista se actualizará y mostrará la siguiente tarjeta disponible, volviendo a mostrar la cuenta atrás. Cuando se terminen las tarjetas, el usuario será devuelto a la pantalla de resumen del mazo.

En caso de que el usuario decida terminar el repaso antes de ver todas las tarjetas, pulsando sobre la flecha de la parte superior o pulsando el botón de volver de su dispositivo, se le

mostrará un diálogo el cuál en caso de ser aceptado, el progreso se guardará y se actualizará en la base de datos, devolviendo posteriormente al usuario a la pantalla de resumen del mazo.

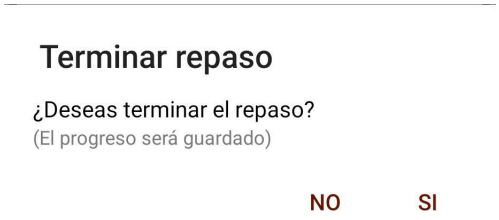

Figura A-10: Diálogo de terminación de repaso